**REFLEXION**

*React to Effects Fast by Learning, Evaluation, and eXtracted InformatiON*



**Methods to extract and apply user / usage profile information to improve the system lifecycle**

**D4.1**

© REFLEXION consortium

Axini – Barco – Océ Technologies – Philips – Siemens ISW – Synerscope – TNO – Yazzoom

**Author(s) :** Corrado Grappiolo, Ardavan Ghaffari, Carmen Bratosin,

Machiel van der Bijl

**Version, Date :** v3.0, 18/06/2018

**Confidentiality :** public

# Abstract

This document describes method(s) how to extract and apply user / usage profile information (models) to improve specific stages of the system lifecycle. We focus on validation of existing and discovery of yet unknown system requirements, and improving verification test coverage, for example by coupling of failure modes.

In the second year, we identified the definition of usage profiling applicable to Reflexion industrial case studies and also the main components of a user profile. In the third year we implemented the methodologies for two main applications – system requirements and testing – and investigated their viability.

# Change log

|  |  |
| --- | --- |
| Version, Date | Description |
| v0.1, 29/05/2016 | initial draft |
| v0.2, 18/07/2016 | internal review version |
| v0.3, 27/07/2016 | reviewed by Mark van Helvoort |
| v0.4, 11/08/2016 | reviewed by Lou Somers |
| v1.0, 15/08/2016 | ITEA first year review version |
| v1.1. 06/07/2017 | update based on second year insights; internal review version |
| v1.2. 07/07/2017 | reviewed by Mark van Helvoort |
| v1.3 10/07/2017 | reviewed by Nils Noorman |
| v2.0 14/07/2017 | ITEA second year review version |
| V2.1 14/06/2018 | Update based on third year insights; internal review version |
| V2.2 15/06/2018 | Reviewed by Stijn Rammeloo |
| V3.0 18/06/2018 | ITEA third year review version (final) |

Table of Contents

[Abstract 2](#_Toc517170281)

[Change log 2](#_Toc517170282)

[1. Introduction 4](#_Toc517170283)

[2. Definitions and classifications 4](#_Toc517170284)

[2.1. Definitions 4](#_Toc517170285)

[2.2. Classification and representations 5](#_Toc517170286)

[3. Methods to extract usage profiles from operational data 8](#_Toc517170287)

[4. Profiles in practice 12](#_Toc517170288)

[4.1 Profiles and requirements 12](#_Toc517170289)

[4.2 Profiles and testing 13](#_Toc517170290)

[5. Case Study: Usage Profiling of Philips Healthcare’s Image-Guided Therapy Machines 14](#_Toc517170291)

[5.1 The data at hand 15](#_Toc517170292)

[5.2 Probabilistic Usage Profiling 17](#_Toc517170293)

[5.2.1 N-Gram modelling 18](#_Toc517170294)

[5.2.2 Unsupervised Learning of Usage Profiles 19](#_Toc517170295)

[5.2.3 The Growing N-Grams Algorithm 21](#_Toc517170296)

[5.2.4 Probabilistic Usage Profiling Validation 22](#_Toc517170297)

[5.3 Hierarchical Usage Profiling 25](#_Toc517170298)

[5.4 Usage Profiling for System Verification / Reliability Testing 26](#_Toc517170299)

[5.4.1 classify-and-test approach via probabilistic usage profiling 27](#_Toc517170300)

[5.4.2 Test Script Generation via Hierarchical Usage Profiling 28](#_Toc517170301)

[6. Case study: Usage profiles for system requirements in the context of Philips MR 32](#_Toc517170302)

[6.1. Data overview 32](#_Toc517170303)

[6.2. Defining the usage 33](#_Toc517170304)

[6.3. From scan parameters to profiles 33](#_Toc517170305)

[7. Conclusions 36](#_Toc517170306)

[Bibliography 38](#_Toc517170307)

# Introduction

In this report we show how to extract profiles (descriptive representations, most of the times in mathematical form, of (classes of) object(s)) from available information sources in the context of high tech industries. Inspired by the available use cases provided by our partners in Reflexion, we define what a profile is and how operational data can be mapped to a profile. During the project, the methods will be used on industrial use cases and the insights will be translated into refining and detailing them.

Profiling represents the act of identifying specific characteristics based on already known information. The characteristics that compose a profile strongly depend on profiling objective. For example, a marketing profile usually focuses on who acquires a system, whereas from a reliability point of view the profile needs to focus on which functionality of the system is most likely used in the customer environment. In Section 2, we provide a classification of the types of profiles and their application.

Once we have determined the profile we are looking for, we need to identify the available information. Technological developments created an ideal environment for collecting more data logged on-line from deployed systems. Logged data has the advantage that it gives an insight into how things are *actually* working, in contrast with questionnaires and interviews often used in market research which elicit the human perception. In Section 3, we apply a typical knowledge discovery process to extract profiles from logged data. Section 4 emphasizes the added value of profiling for two specific applications: system requirements and system testing. In complex evolutionary systems discrepancies exist between the explicit end user requirements, the system specifications (e.g. defining allowed usage area), the working latitudes of the system (where and how does it really break down), and normal and exceptional usage. Profiling closes the loop between deployments and new developments. For example, profiling reveals the most and the least used functionalities. This information can be translated on improving requirements for future developments or into more relevant reliability tests. Section 5 describes how usage profiling can be applied to a case study, namely modelling the usage of x-ray systems of Philips Healthcare’s Image Guided Therapy. The Section outlines how the logged data is processed for the profiling purposes. It describes two approaches for usage profiling – one leveraging on the sequence of events and the other one leveraging on the hierarchical structure in the data – together with a possible way to use them for system verification testing – one for the creation of a cooperative tool machine-expert to select data to be tested and one that attempt the generation of automated test script by combining usage profiles with model-based testing. Section 6, instead, centres its attention to another case study, namely modelling the usage of magnetic resonance imaging systems of Philips Healthcare’s MR division. For this case study we attempted the task of usage profiling via process mining and domain-specific languages. Finally, Section 7 concludes this document.

# Definitions and classifications

### 2.1. Definitions

Profiling, in general, has many meanings thus to become more specific we focus on the following definition:

*A* ***usage profile*** *is represented by the correlations and patterns, temporal and a-temporal, found in the collected log-files* *from the installed base.*

Our definition restricts the usage profiles to the data collected from the installed systems and excludes profiles that can be created based on design or other ways of collecting data. In this manner, we determine our focus to be on log-files. Note that this definition will not exclude the possibility of enriching the log files data with domain knowledge that will help better understanding or better structuring of the data.

Additional, we introduce, based on [1] the following profile categories:

1. *Group profile* – aggregates information from multiple log-files that can span over multiple customers, users, systems etc. A group profile can be of two sub-types:
   1. A *community* is a predefined group (e.g. hospitals in the Netherlands)
   2. A *category* is a group resulted from profiling, e.g., as part of a clustering step

Additionally a group profile can be characterized as: *distributive* when the profile identifies a group of which all members share all the attributes of the group, e.g., all users of a medical system are trained radiologists; for each of its members the profile is also a personalized profile, or *non-distributive* when it identifies a group of which not all members share all the attributes of the group's profile, e.g., 75% people in neighborhood A have high income and 67% are between 35-40 years old.

1. *Personalized profile* – focuses on only one log-file, one customer or one system, offering an in-depth view of the profiled subject.

Most of the time, a usage profile is a non-distributive group profile that is a combination between a community and a category. For example, in the context of Philips IGT, we create usage profiles that are specific for each anatomical region (the community aspect) that we can further cluster into different type of usage based on data analysis algorithms (e.g. data analysis techniques reveal that there are two types of profiles: 1) only imaging and 2) imaging is combined with hardware movements).

### 2.2. Classification **and** representations

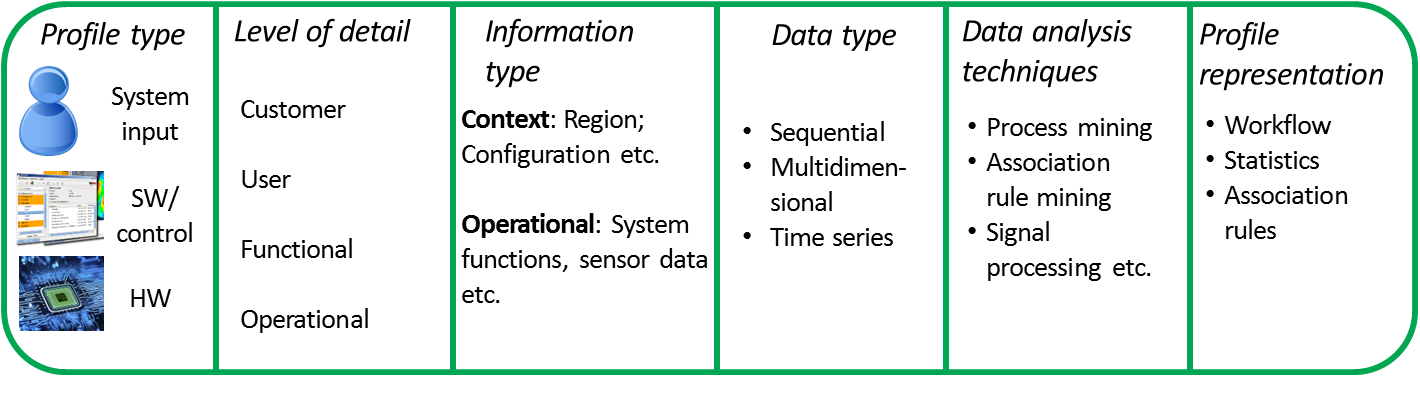


Figure 1: Main components of a usage profile

When constructing a profile, we identified a number of six components that need to be described: profile type, level of detail, information type, data type, data analysis techniques and profile representation. In the following subsections, we will detail each of these components.

#### Profile type

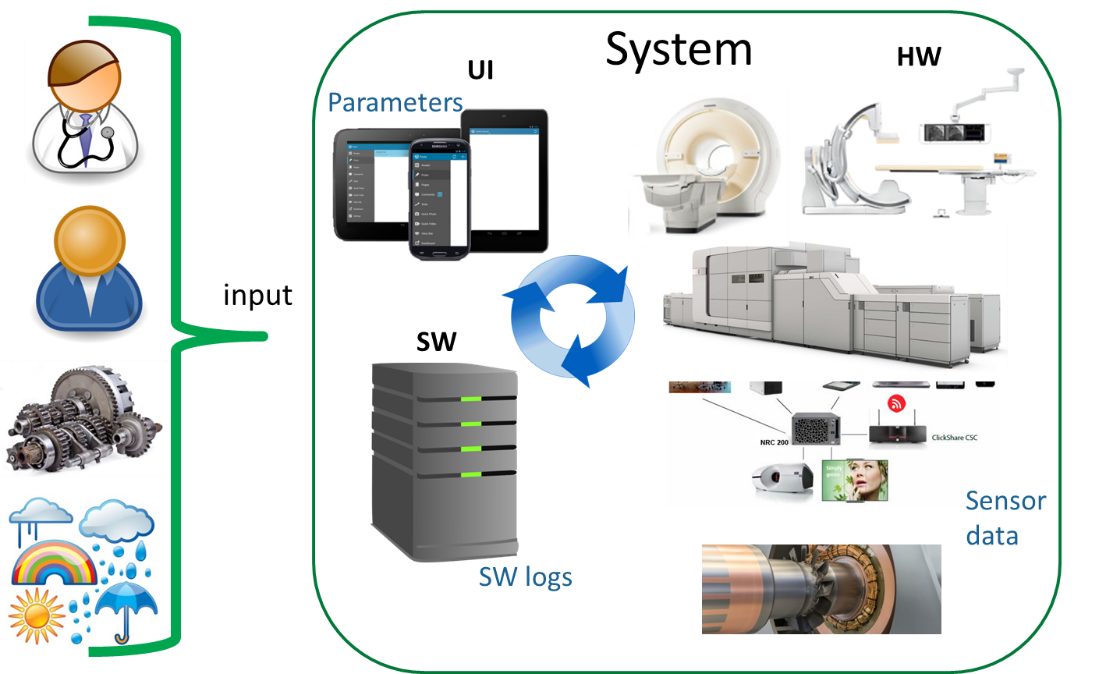


Figure 2: A system and its context

The profile type is given by what part of the system is of interest. In Figure 2, we show that the system can be decomposed in multiple parts: the input that it receives from, e.g., users, other systems or the environment, its user interface (UI) that allows direct connection between users and the system, the software that usually defined high level workflows and it coordinates the hardware equipment, and the actual hardware.

In certain applications, e.g. system testing, the system is seen as a black box and thus only its input is of interest to be profiled. We called this profile type *system input.*

On the other hand, for applications such as root-cause analysis, we need to look and understand how the system is designed and operates, i.e. the system is seen as a white/grey box. In this case, inspired by the Reflexion applications, we identify two additional profile types: *software or control profile* and *hardware profile.* The difference is given by the element of the system, as also the name suggests, that we want to focus on.

Once we establish which profile type we are interested in, the source of information for the profile can also be identified. For example, if the system input relates to users’ actions then logging of the UI parameters should be considered.

#### Level of detail

Profiling can be seen as slicing the usage space along certain dimensions or levels. Based on a literature review [2] [3], we identified five major levels (see Figure 3): customer, user, system-mode, functional and operational.

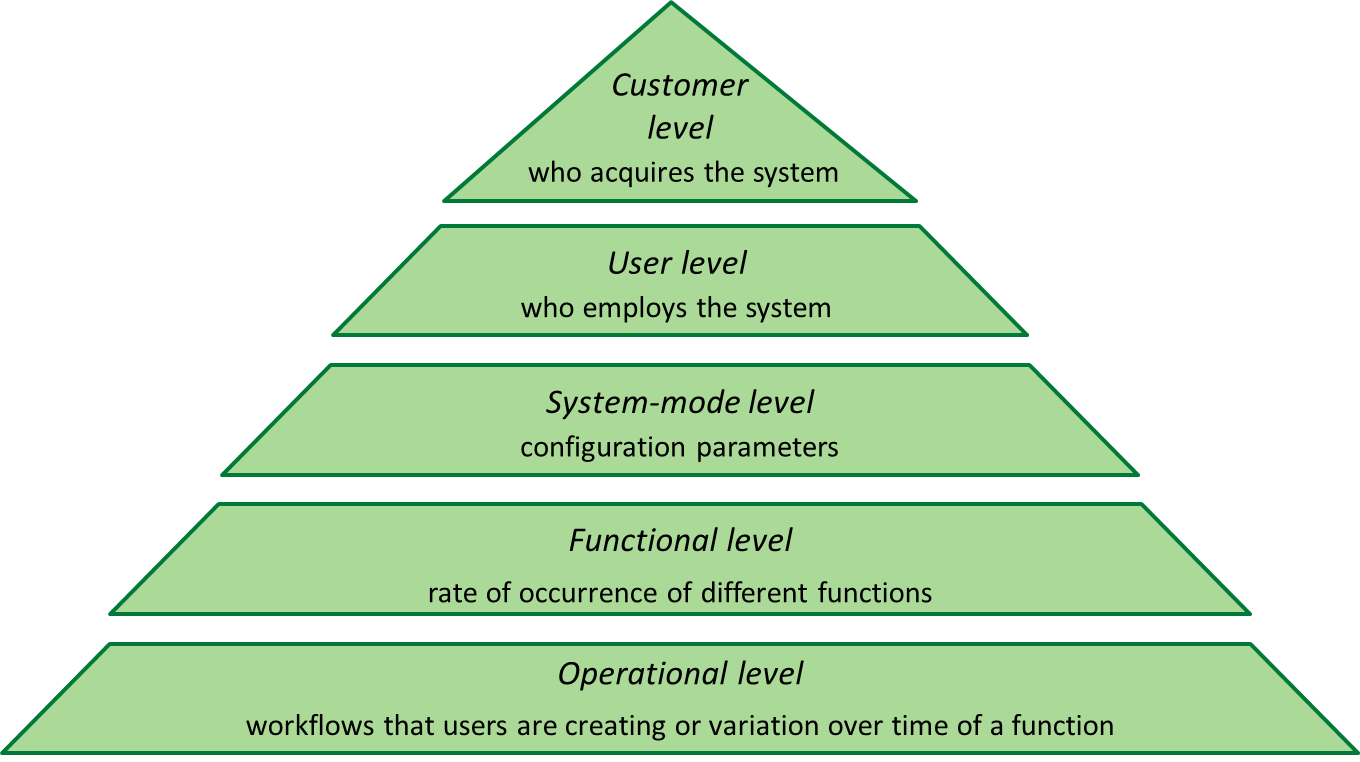


Figure 3: Level of details for usage profiles

At *customer level*, we specify who acquires the system by characteristics such as country, type of business, specialty, type of preferred products, etc*.* Thislevelisrelevant in understanding the demographics of the potential buyer of the system but also its interests and needs.

*User level* sketches who employs the system by features such as age, profession, level of education, preferences, etc.

*System-mode level* focuses on the configuration of the system, e.g., which licenses are used, what type of functionalities are available or what type of hardware is used.

*Functional level* reflects the rate of occurrence of different functions. It captures the system usage in a customer environment. Usually expressed as statistics over function usage, it is a static view over system usage. Therefore, we can capture here also relations between functions in the sense that what is the probability of a function A to be employed if function B is employed.

*Operational level* is the process grasping of the workflows that users create by using the system or the variation over time of a system function. It bridges the customer environment and the system functions. At this level, by modelling the workflows created by users or the behavior over time of a certain function, we describe the dynamics of a system. Due to its dynamic nature, the profile is usually represented by graph-like structure (state-machines, Markov chains, workflow nets etc.) or time-series.

As expressed in [3], most of the time the levels are combined to give a better overview. For example, combining the customer and functional levels facilitates targeted marketing or new system proposals.

#### Information type

When creating a profile, we need to specify what information is of interest for describing different levels of detail. We split the information into two types: *context information* that is either static or slow changing, and *operational information* that is highly dynamic.

The exact content and significance of the information is highly dependent on the domain. For example, in the healthcare domain, context information is if the user of the machine is a doctor or a nurse, where for wind turbine if the turbine is on land or on the sea. Same for operational information, we need to understand the actual system decomposition to specify the content.

#### Data type

The next component to be taken into account is what *type of data* is available. We identified three major types:

1. *Sequential data* that is usually event data as in software or workflows logs. This data is often structured into: timestamp, event name and additional meta-information about the event.
2. *Multidimensional data* that captures instantiation of multiple features in a certain time-frame. Such data can be considered for, e.g., affinity analysis [4].
3. *Time-series data* is usually sensor data that represents one particular evolution in time of a numerical variable.

The data type determines the type of pre-processing and analysis that can be employed, the tools and software needed or the expertise of the person to perform the analysis.

#### Data analysis techniques

The fifth component is highly correlated to the previous one since the data type determines in a high degree the type of analysis one can employ. Examples of data analysis methods are: machine and data mining, associative rules mining, neural networks, signal processing, process mining, statistical analysis, etc.

Moreover, a particular data analysis method might require input data to have very particular formats. Additional requirements may be imposed such that the data to be labelled if supervised learning techniques are used [4], or the minimum amount of data needed such that the result to be significant.

#### Profile representation

Once the analysis is done, it is important to specify in which format the profile would be represented. Profiles are usually represented via graph-like representations, which can be easily read and interpreted by domain experts.

# Methods to extract usage profiles from operational data

In the past, profiling was mostly done through interviews, questionnaires or customer visits. The possibility to gather data on-line from the customer environment allows a better glimpse into the actual usage of a system. Multiple papers, such as [2] [5] [6] [7] [8] [9], showed that it is beneficial to use logged data in order to extract better usage information from deployed systems, and at the same time emphasize the need for better logging.

As stated in [10], currently available logs are created mostly for debugging purposes. This creates a challenge due to lack of information about users and environment, or the high granularity of details about internal system states. Other issues are the lack of standardization, log distribution over multiple files/databases, misinterpretation of log entries. Alspaugh et al. [11] propose the following list for improving logging: “capture high-level user actions, capture provenance of all events, observe intermediate user actions, obtain the analyzed data's metadata and statistics, work towards log standardization, collect user goals and feedback”.

Once we retrieve the operational data (logs created during system usage) we can follow the flow from Figure 4 to first determine the components of our profile as described in the previous section. We always start by fixing the application for which we aim to employ the profile. The reason is that the application imposes a lot of constraints on the subsequent components. For example, in one of the Reflexion cases that focuses on profiling for testing we profile the *system input* using mainly *functional and operational level* where the type of information is the order in which *system functions* are employed by a user in the *context* of a specific *medical procedure*. Such data has a sequential nature that can be analyzed by both statistical methods and process mining. The profile representation is *a workflow augmented with statistical information*.

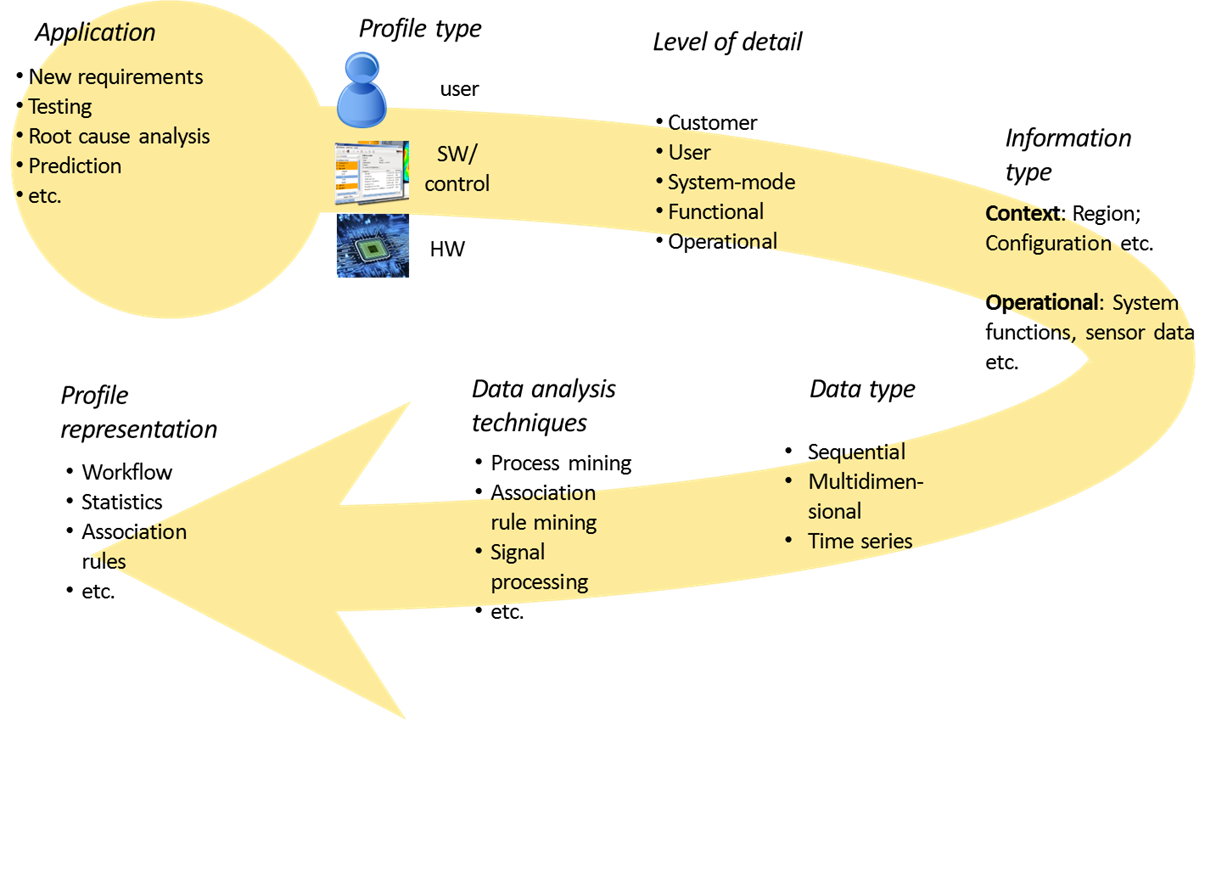


Figure 4: Flow to determine the components of a profile

Once the application and the profile components are chosen, the process of extracting a profile follows a typical knowledge discovery process. Figure 5 presents a typical such process as presented initially by Fayyad et al. [12] and generalized in [13] based on a literature survey.

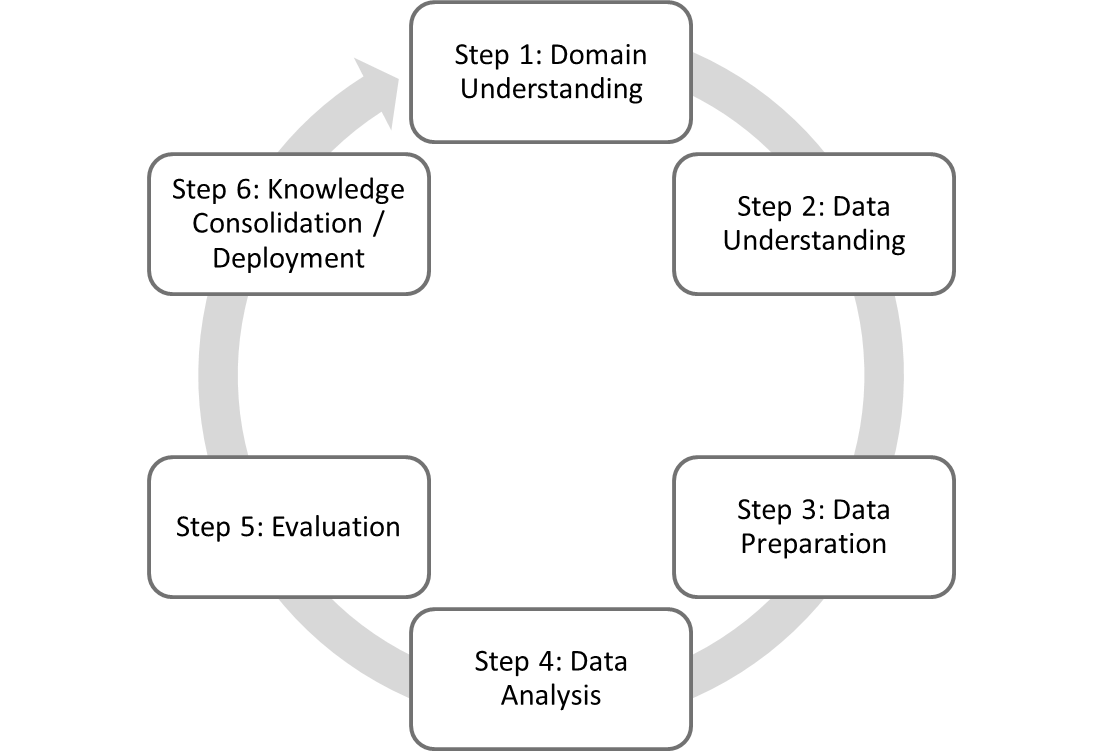


Figure 5: A 6-steps knowledge discovery process.

In the context of profiling, the outcome of step 1 of Figure 5 relates to the first three components of a profile: the profile type, level of details and the information type.

In step 2 we need to select the information sources and their relationship with the characteristics we are looking for. For example, if a customer level is of interest customer relations management databases should be considered as a source of information, for user profiles identities of users must be available through sign-in data.

For data preparation (step 3), apart from filtering of noise and selection of the important attributes, one important step is translating the available data from low-level information to the abstraction level relevant for the chosen profile. Figure 6 presents four possible abstraction levels and a possible mapping between low level events and higher abstraction levels. Such mapping is most of the time not trivial, since low-level events can be mapped to multiple high-level functions. For example, Gunther et al. [5] present a case from Philips Healthcare where they discover as a common sequence *Patient Preparation*, *Procedure Left Coronary*, *Procedure Abdomen*, and *Patient Completion*. The low-level events such as *Start Fluoroscopy* and *Stop Fluoroscopy* can be both parts of *Procedure Left Coronary* and *Procedure Abdomen*.

In this case, context information and domain knowledge may facilitate the translation of the log into a higher abstraction. To achieve the translation, two approaches are feasible as found in the literature:

1. A data driven approach uses classification or clustering algorithms to define the mapping. In [14], Bose et al. proposes an iterative method to identify common events patterns and to derive log abstractions based on them.
2. Usage of requirements documents to define low-level to high-level mapping that is further used to define a new abstract log.

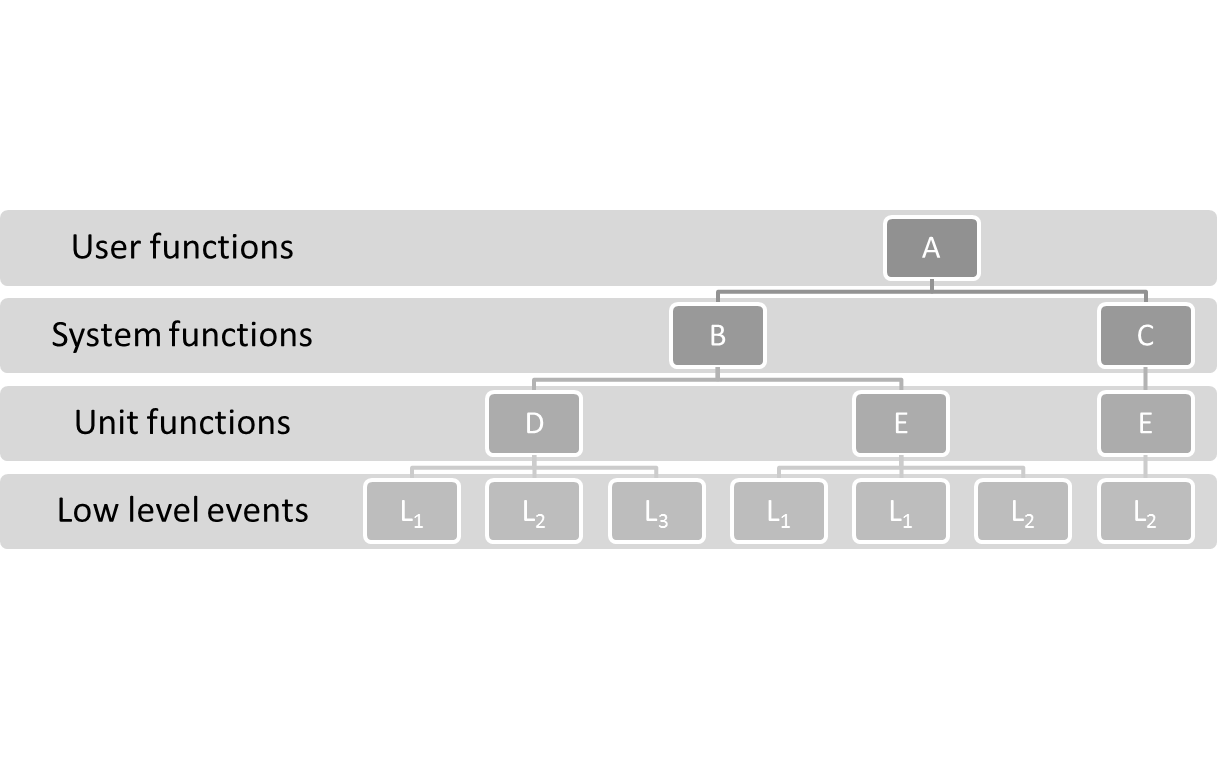


Figure 6: Log granularity and abstractions levels. The profiling is usually required from the point of view of the user thus available logged data contains information about low-level events.

Once data is available and at the correct abstraction level, we can identify two important aspects for profile discovery (step 4 of Figure 5):

1. *Data driven or model discovery* thatassumes that the data is in the lead and contains all the necessary information to answer the question. Machine learning [15] and data mining [16] are techniques which leverage on data to retrieve statistical patterns. Other techniques are for instance process mining [17] or association rules learning [18]. The main advantage of these approaches is that they would capture usage behaviours which could deviate from what domain expert assume. The main disadvantage is that logged data, being it generated from other purposes e.g. system debugging, is often polluted by redundant information and noise which need to be filtered, cleaned and organised. This can however be limited by relying on domain knowledge to add extra information to the logging that can help in, for example, creating data hierarchies [19]. For the latter one, domain experts should be able to overrule the trivial results and the knowledge can be used to re-iterate the analysis until better results are created.
2. *Model driven or model fitting* starts with considering an a-priori model for the profile based on, for example, specification or design documents, experience, domain knowledge. The data is then used to detect parameters of the model, or to deduce probabilities of certain paths in the system, add timing information etc. For example, [20] uses log data to add history to a model that is then used further on for recommendations. Such a method adds value especially when the amount of features is too big. On the other hand, it might overlook deviations from the actual behaviour of the system or can discard valuable information because it will not fit the assumed model.

The two aspects can be combined in an iterative process to derive the most important features that compose the desired profile.

The last two steps are heavily dependent on the target of profiling. In these steps, visualization techniques and domain specialists are required to interpret and draw conclusions. In the next section, we will elaborate on the possible applications of profiling and their added value.

# Profiles in practice

By providing new information, operational data helps to close the loop between deployment and new developments. This becomes far more important when new products are increments of existing released products. Understanding the customer value of the increment via operational data in the usage profile, for example, guides and prioritizes the functionalities for the new release.

We identified 4 applications grouped into 2 groups inspired by industrial case studies present in Reflexion:

* 1. System requirements
     1. Dashboards
     2. Machine set-up
  2. Testing
     1. System testing based on user workflows
     2. Control testing based on previous anomalies

For each of these applications, we aim at the end of the project to provide a detailed description of which components (see Section 2) are relevant and the reasoning on how to choose a certain component over another. Such a description will help the industrial partners into determining the requirements and the added value of profiling and for the SME’s to adapt their software offer towards the industrial needs.

Following we will give a high-level description of the possible impact of profile for the three main categories identified.

4.1 Profiles and requirements

Requirement engineering concerns identifying, documenting, negotiating, and managing the desired properties and constraints of systems, as well as the goals to be achieved by the system, and the assumptions about the environment [21]. Traditionally capturing requirements involves users through customer visits, user interviews, and workshops. The question arises how practitioners can use the “implicit feedback” [22] of users gathered through usage data, logs, and interaction traces to integrate it into the process of defining the next release. A proposal has been made by [22] to collect this information and to present it to developers such that they can derive the user needs based on actual usage. The authors predict a “paradigm shift in requirements engineering and software evolution towards data-driven user-centered development, prioritization, planning, and management of requirements”.

The impact of profiling for requirements engineering is two-fold: requirements prioritization and creation of new requirements. Requirement prioritization is the process of defining in which order the functionalities will be implemented or released. The reasoning is to minimize costs and maximize impact.

New requirements may emerge by understanding the way users are using their software and their reasoning. For instance, for the Philips Healthcare’s MR case study (see Section 6), we use different data mining and knowledge discovery techniques to define market segmentation and, to deduce user intentions based on how users set the settings and parameters of the system. That allows us to guide system architects into understanding the context in which their product is used. This information can define the boundaries of the new releases of the products.

4.2 Profiles and testing

Model-based testing makes a model before and during system design and implementation. The state space of complex systems is often very large. Additionally, there is a limited amount of time allocated for testing. This means that during design and more specifically during testing we have to make choices. As a result, we can only validate specific parts of the design and implementation space. A priori it is difficult to know what part of the state-space is more important than other parts. However, users of software and hardware systems can provide this information: by definition, important features are the ones that are used.

When combined with user profiles, individual system tests can be prioritized and placed in a sequence that optimizes coverage for real usage. The combination of this approach with system modelling, system simulation (record/playback), and automated derivation of test scripts, or model-based testing and model checking, reduces the chance that user-relevant failures are not found during the in-house test phases. On basis of field data system modelling and testing will be improved. One of the options for this is fault modelling. A fault model specifies behaviour that is forbidden to happen. With field data, we can sharpen this model, monitor in the field if the system exhibits dangerous behaviour, and prevent user risks. This is a combination of fault modelling and passive testing.

In Reflexion, we use the field data in the modelling and testing phase in order to close the loop between design and production. We call this endless profiling. Endless profiling will be used for root cause analysis, identification of user risks and fault detection. We will do this via modelling, simulation (record/playback), model-checking and model-based testing. The first thing to address is what field data to make available (requirements).

The available logged data contains low-level events that do not directly translate to high-level functions. A technique we will use to address this problem is action refinement to relate tracing/logging to interface/API actions [23]. We will then generate usage models and test guidelines for endless testing. Topics for this phase are fault modelling, test-case generation and user modelling/profiling. A fault model specifies behaviour that is forbidden to happen. With field data, we can sharpen this model and we can monitor in the field if the system enters dangerous behaviour and prevent user risks. This combines fault modelling and passive testing. Based on the field data we can make a model of typical usage. By taking the product of this user model with the design model we can reduce the state space. Furthermore, with the user data we can enhance our models and identify model-parts that are more important than others and use this for example in test-case generation and exploratory testing.

1. Case Study: Usage Profiling of Philips Healthcare’s Image-Guided Therapy Machines

This Section delves into a practical application of usage profiling for modelling and subsequently for system verification testing in a cyber-physical system, specifically Philips Healthcare’s Image-Guided Therapy machines (IGT). Philips IGT is a business unit of Philips Healthcare which creates interventional x-ray machines. These machines are used in catheterisation laboratories and hybrid operation rooms to help doctors in minimally invasive procedures. Allura Xper X-ray systems (Figure 7) were used in this research.



Figure 7: the Allura Xper x-ray system

These systems are very complex, as they provide a lot of functionality to the users and not all functionalities are used in the same way. Some are used very frequently, and some are rarely used. This information (usage of the system in the field) is available from logged data, however testers do not have a way to incorporate it in their tests yet. Instead, risk-based testing is done based on experience and field visits.

Section 5.1 briefly describes how the log data is being processed for our case study. We also considered two modelling approaches. The first approach relies on machine learning techniques to cluster the input logs into classes of profiles, and subsequently relies on N-Grams – a modelling technique proper of Natural Language Processing (NLP) – to represent the classes of profiles as conditional-probability graphs. This approach – which we will hereafter refer to as the probabilistic approach – attempts to leverage on the temporal dependencies of usage actions to retrieve different usage behaviours. It is described in Section 5.2.

The second approach is instead centred on the extraction of a single hierarchical relationship of statistical distribution of usage actions. The different usage profiles are then retrieved by traversing the hierarchy depending on its multimodal distributions. The hierarchy is manually-designed and accurately represents the nested dependencies of events, sub-events, and phases of medical procedures. This approach – which we will hereafter refer to as the statistical approach – is presented in Section 5.3.

Currently, system verification tests are conducted by executing manually written scripts. Section 5.4 describes how usage profiling can be used to improve system verification testing. Based on the two modelling approaches we took two standpoints. The first one aims at using the probabilistic modelling approach to facilitate the script selection task to be performed by the tester in a cooperative environment, where the expert-tester can also provide feedback to the modelling algorithm in order to better perform its task. The other standpoint leverages on the hierarchical approach to generate artificial test scripts.

5.1 The data at hand

It is often common, beyond the case studies considered in Reflexion, that the data at hand is not perfect. As we said previously, machine logs are often generated with the purpose of debugging, this means that a lot of redundant information needs to be filtered. This can hardly be achieved without the help of domain experts.

In the specific case study at IGT, domain experts provided us with a list of relevant event keywords. These keywords – also called actions hereafter – were selected in order to capture events which directly relate to the usage of the machine.

During a day, multiple patients are diagnosed and/or treated on an individual system. An examination typically consists of a patient preparation phase in which the patient enters the room, patient administration is done, and everything will be prepared for examination. Then several medical procedures are performed sequentially. After the procedures are performed, the patient leaves the room, the generated images are reviewed, and the x-ray machine is prepared for the next patient. This process of patient preparation, examination and completion is schematically displayed in Figure 8.

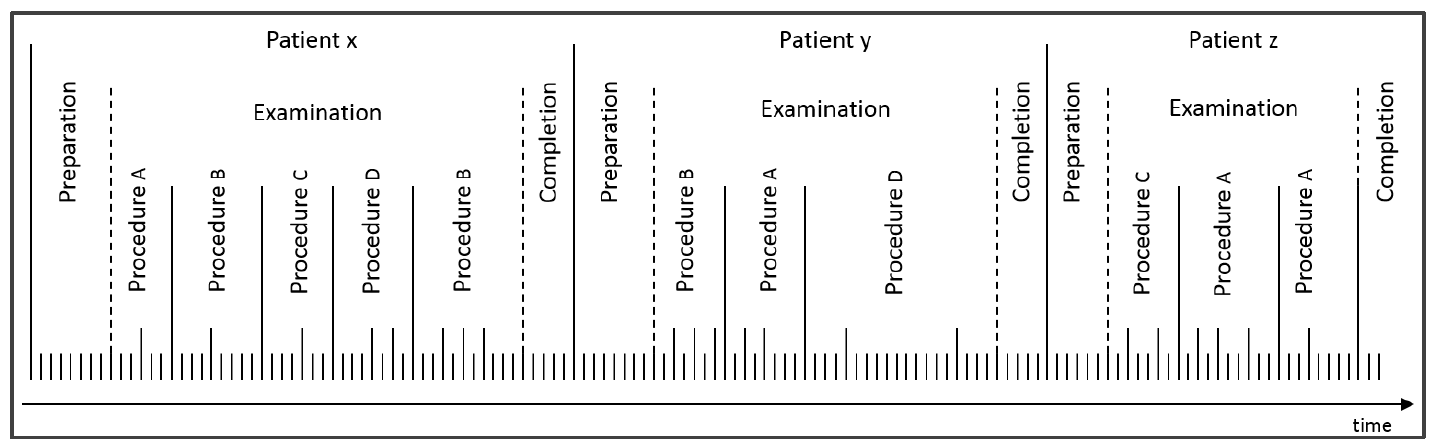


Figure 8: Schematic representation of the patient preparation phase,   
examination phase and patient completion phase over time.

The preparation and completion phases are fixed among all test cases. Therefore, these two phases are not considered for the modelling task.

**The Bolus-chase Case Study**

The case study considered for the probabilistic approach to user modelling of Section 5.2 is the bolus-chase procedure. Moreover, it was decided not to focus on the single duration of the events, rather on the chain of events following each other. The remainder of this section describes bolus-chase and situates our task of data gathering and cleaning.

The bolus-chase procedure allows doctors to visualize the entire vessel structure of the legs. Before actually starting the procedure, it is first checked whether the table can freely move from the patient’s abdomen to the feet without any collisions. Then a ‘contrast run’ is performed, in which a bolus of contrast agent is injected, directly followed by image acquisition combined with table movement in order to follow the bolus flowing towards the feet of the patient. After exposure is done, the table is moved back towards the start-position. Optionally a ‘mask run’ is performed. This mask run is acquired with the same exposure settings and movement profile as recorded during the first run, but without injection.

While this in general describes the workflow of a bolus-chase procedure, there are probably no procedures identical due to hospitals having different workflows, different doctors, different patients, etc. All reasons of having different versions of the basic procedure described above. This makes it difficult to model the bolus-chase procedure as a whole. Based on inputs from experts it was decided to cut the logging in chunks delimited by x-ray events, and to only start analysing as of the second chunk (sub-procedure). This is because the edges of consecutive procedures are often overlapping. It might be that images of patient x are still being reviewed while patient y is already placed in the examination room. To eliminate noise from previous procedures, it has been decided to only start analysing as of the second sub-procedure. Figure 9 depicts such filtering process.

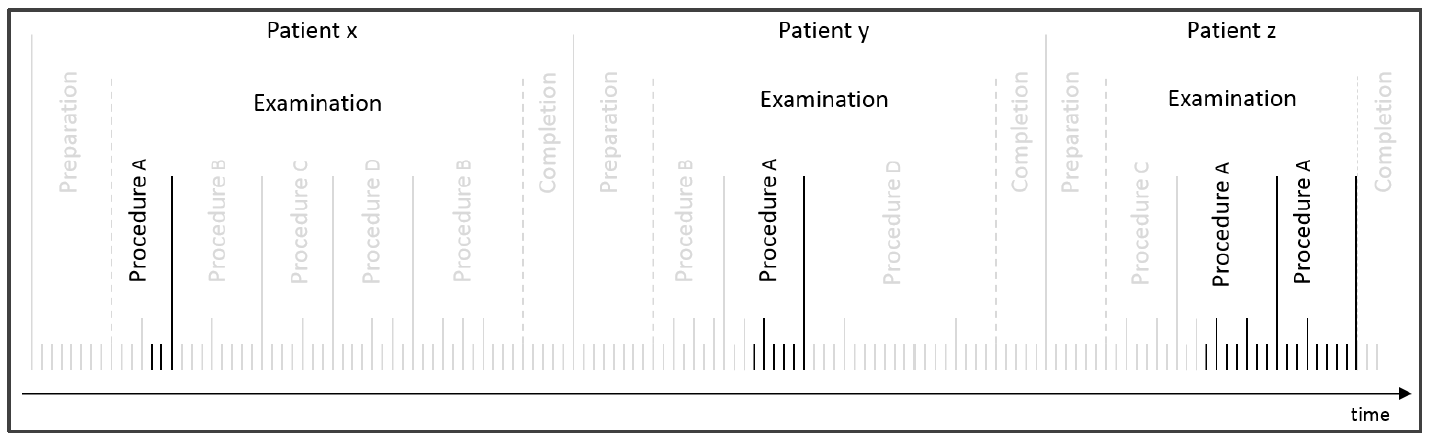


Figure 9: Schematic representation how procedures consist of a chain of sub-procedures, where similar sub-procedures have the same colour labelling.

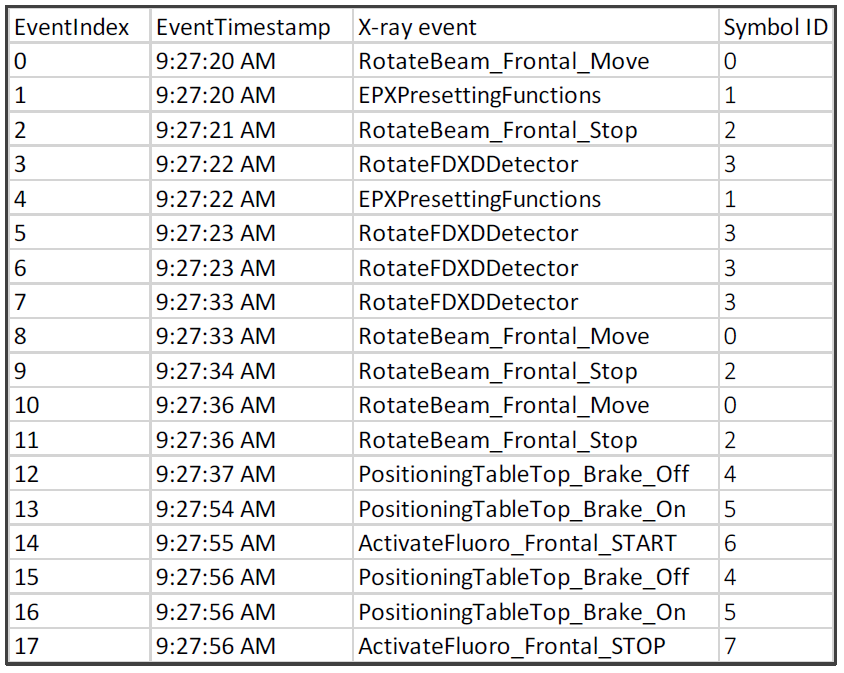
What we are left with is a set of log sequences. An index is then generated, so that each log event would be mapped to an integer value. Figure 10 depicts an excerpt of a bolus-chase sequence and its indexing. For our usage profiling task, we solely focus on the first column, “EventIndex”, and the last column, “Symbol ID”. The former column allows to solely focus on the chain of events and ignore the duration of each events, whilst the latter column allows for the representation of a single log in a tokenised fashion of strings of symbols. The indexing of events is extremely important, since now we can focus on string clustering, which allows us to rely on disparate machine learning techniques for unsupervised learning.

Figure 10: Excerpt of a bolus-chase sequence and relative indexing

5.2 Probabilistic Usage Profiling

Prior to attempt modelling of a single usage profile, we need to overcome the obstacle to identify how many profiles exist. Unfortunately, relying on domain expert knowledge could be detrimental, since they would certainly be biased towards specific usage characteristics. Moreover, one expert assumption might be counter-argued by another expert’s. Nevertheless, even if we find ourselves in a situation in which experts agree on some assumptions, we might be unable to rely on their knowledge to identify corner cases of usage. Such problems relate quite well to the gap existing between system specification (domain-expert side) and system usage (in-field side).

In machine learning terminology, the problem of identifying how many usage profiles exist without any a-priori knowledge corresponds to an *unsupervised learning* problem. The approach is to *cluster* together profiles which are alike – maximise within-cluster similarity – and keep non-alike ones in different clusters – minimise between-cluster similarity. Once usages are clustered correctly, the subsequent task of usage modelling would be trivial, or at least easier to achieve.

In order to attempt the usage modelling of IGT machines in their operational domain we decided to focus on probabilistic modelling, for several reasons:

* humans can easily understand probabilities;
* probabilities describing the transition between similar events/actions can be grouped together resulting into generalisations of usage;
* the aggregated probabilities could be organised as graphs, leading to a vast plethora of graph manipulation operations, e.g. partitioning, walking, etc.
* algorithms outputting probabilities – i.e. floating-point values – can trivially be combined with other computational algorithms which could further enhance the modelling task;

In order to approach our usage modelling task and identify the algorithmic techniques needed we adopted the following metaphor: a machine log is a text, written in machine language, describing its operation. Therefore, similar logs would tell similar stories and should be grouped together. By identifying and modelling different types of stories we would achieve our usage modelling task.

Through that metaphor we can now rely on a vast plethora of computational techniques proper of Natural Language Processing (NLP) [24]. This also means that several NLP approaches to language modelling exist. It is in fact possible to find studies relying on e.g. the consolidated N-gram approach [25] or novel work centred on deep learning, recurrent neural networks and their flavours [26]. Intuitively, the aim of the modelling task – for instance identify and describe usages or generate new by mimicking them – the different stages during which domain experts should provide their contribution, or even the degree of “blackness” of the modelling approach, would ultimately allow the identification of specific computational techniques.

However, NLP techniques are studied with the aim to model *natural* language, as opposed to our aim to model *machine* language. The research conducted within Reflexion also contributes to shed some light on whether and how easily NLP techniques can be extended to an industrial domain.

5.2.1 N-Gram modelling

N-grams [25, 24] are among the most widely used method for language modelling. An N-gram is a sequence of N words: a 2-gram (or bigram) is a two-word sequence of words like “please turn”, “turn your”, or ”your homework”, and a 3-gram (or trigram) is a three-word sequence of words like “please turn your”, or “turn your homework” [24].

N-grams rely on the statistical frequencies of sequences of words in order to estimate probabilities of observing a word given a history of previously observed words. Once the probabilities are estimated, N-grams can be used for instance to estimate the probability of a next word or of whole sequences, they can be traversed in order to generate sentences etc. To estimate probabilities N-grams rely on the chain rule of probabilities, whilst for the traversing of N-grams the technique used is random walk [25, 24].

There are several pitfalls related to the transitioning from frequencies to probabilities, such as being able to calculate probabilities of sequences or even symbols which have not being observed during the training phase. Likewise, several techniques have been proposed, such as interpolation or smoothing [25, 24]. In our case study we relied on Kneser-Ney interpolation and backoff [27].

To build N-grams, an input corpus of sentences is required. In our case study, then, N-Grams are modelled only after the string clustering task is performed.

5.2.2 Unsupervised Learning of Usage Profiles

String clustering can be performed in many desperate ways. However, since the approach adopted in unsupervised learning is maximisation of within cluster similarity and minimisation of between cluster similarity, a similarity measure is required.

We here centred our attention on three techniques to compute string similarity:

1. string-tailored similarity measures;
2. word embeddings, a recently proposed NLP technique capable transforming data from the string-space to a n-dimensional vector space [28];
3. growing N-Grams, a novel approach to string clustering researched within Reflexion, which relies on the construction of N-grams online and set difference of sequences.

**String-tailored similarity measures**

Our research relied on three well-established measures for string-based similarity, namely the Levenshtein distance, its normalised version, and the Jaro distance.

The Levenshtein Distance (LD) [29] calculates the minimal number of insertions, deletions or substitutions required to transform a source string into a target string. The main shortcoming of LD is that it only focusses on the differences between two strings and not taking any similarities into account. Marzal and Vidal [30] found a solution to this by taking additionally to the distance between two strings, also the maximal string length into account (Normalised Levenshtein Distance, NLD). NLD results in a value in the interval [0 ,1], where 0 means the strings are equal whereas 1 means there are no characters in common.

Finally, the Jaro Distance [31], which is less straightforward than the previous two, has the advantage that it leverages on a matching score functioning as a threshold below which two strings are considered completely different (value 1). In case two strings pass the matching threshold, the actual string distance is calculated.

Other string similarity measures, not considered in our study but which could be worthy being investigated, are for instance weighted Levenshtein distance [32]. For an overview of possible string similarity measures see [33].

**Word Embeddings**

Word embeddings are a way to transform words in text to numerical vectors so that they can be analysed by standard machine learning algorithms that require vectors as numerical input.

A common and most basic type of word embedding is called one-hot encoding. One-hot encoding represents a word in the text by a vector of the size of the vocabulary, where only the entry corresponding to the word is a one and all the other entries are zero.

A major problem with one-hot encoding is that there is no way to represent the similarity between words. In any given corpus, one would expect words such as (cat, dog), (knife, spoon), and so on to have some similarity. The same would apply to our usage profiling domain: events such as (Rotation, Angulation) for C-arm movement and (Fluoroscopy, Exposure) for x-ray events.

Similarity between vectors is computed using the dot product, which is the sum of elementwise multiplication between vector elements. In the case of one-hot encoded vectors, the dot product between any two words in a corpus is always zero.

There exist several word-embedding techniques, such as latent semantic analysis [34] and word2vec [35]. For our case study we focused on the doc2vec embedding technique, which algorithmic details can be found in [28].

**Hierarchical Clustering**

The purpose of clustering is to discovers unknown subgroups in unlabelled data so that the “within cluster similarity” is maximised and the “between cluster similarity” is minimised [36]. Popular clustering methods are connectivity methods, centroid methods, distribution methods, and density methods [37] [38]. In this Section we only outline hierarchical clustering – a connectivity method – which is the algorithm we considered in our study.

Connectivity methods cluster data based on distance connectivity; objects closer in data space exhibit more similarity to each other than objects lying further away. These methods can be either agglomerative (down-top) or divisive (top-down) [39]. Agglomerative algorithms begin with classifying all objects into separate clusters and merge them in successively larger clusters. Divisive algorithms begin with the whole dataset in one big cluster and proceed to partition it into successively smaller clusters. Examples of these methods are hierarchical clustering and its variants. These methods are typical ‘hard’ clustering methods, since all objects of the dataset are clustered. An advantage of connectivity methods is that they only require a matrix with the pairwise similarities between all objects of the dataset. Another advantage is that they can easily be represented in a dendrogram. Figure 11 depicts an example of a dendrogram. The horizontal axis contains the objects whereas the vertical axis represents similarity among objects. The lower objects are merged, the more similar they are. Based on this dendrogram, the number of clusters can be defined by ‘cutting’ the tree at a certain level, as is shown in Figure 11. The lower the cut, the more similarity within cluster but also the higher the number of clusters.

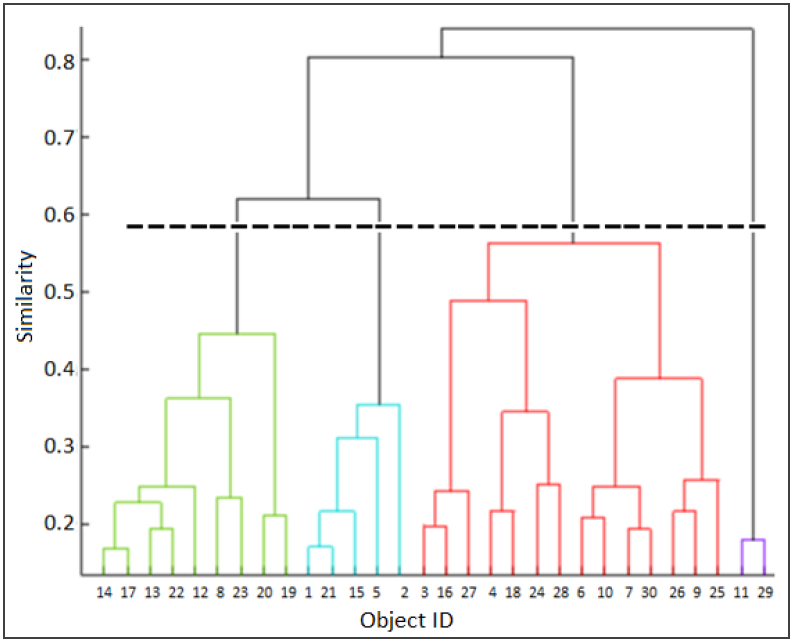


Figure 11: An example of a dendrogram resulting from a connectivity clustering algorithm. The horizontal axis contains the objects which are ordered based on their pairwise similarities to avoid crossing lines. The vertical axis denotes similarity within clusters

Possibly, the most popular method in determining the number of clusters for hierarchical clustering is the elbow criterion. The success of this algorithm is because of its interpretability and straightforwardness [39] [40] [41]. The intuition of the elbow rule is that the optimal number of clusters is such that adding another cluster does not add sufficient information. With respect to Figure 11, the elbow would correspond to the horizontal dashed line. The percentage of variance of the fictional data of Figure 11 is depicted in Figure 12.



Figure 12: Variance unexplained for the data in Figure 11. The elbow is clearly visible for k=3

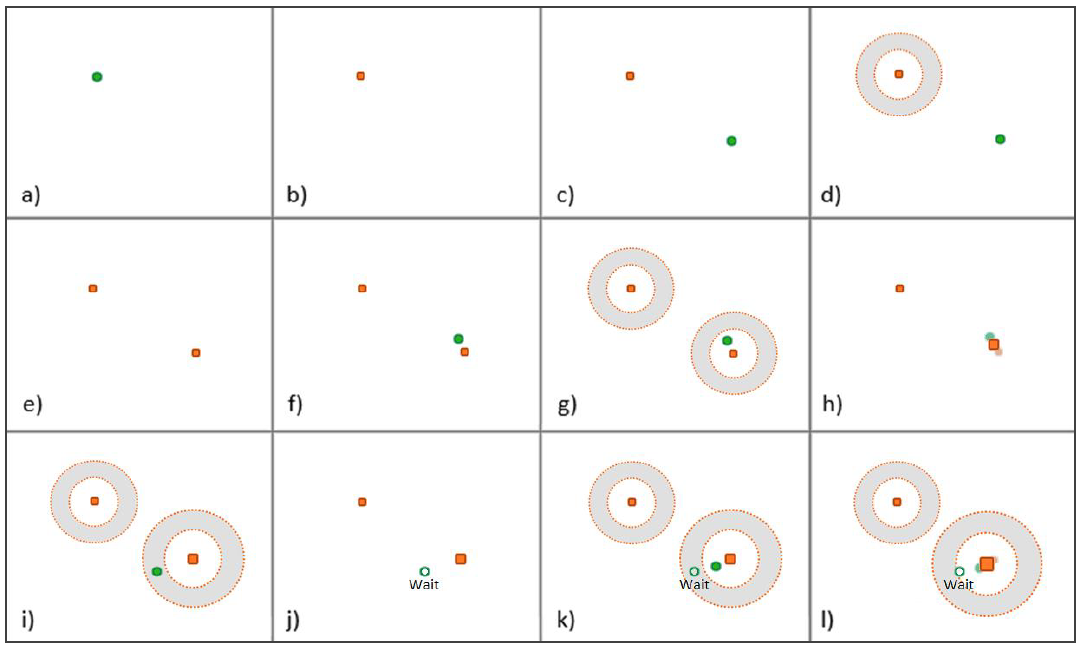
There are several ways to define similarity between two clusters. The two most popular algorithms are single link [42] and complete link [43]. Single link merges the clusters whose distance between their closest objects is smallest, whilst complete link algorithms merge clusters whose distance between their most distant objects is smallest. In general, single link algorithms generate extensive clusters whereas complete link algorithms generate compact clusters. The major drawback of these kind of methods is that they are greedy since they cluster minimum/maximum distances at each step without taking successive steps into account.

5.2.3 The Growing N-Grams Algorithm

Normally the corpus is known when estimating an N-gram model. We designed a new clustering algorithm that creates and grows N-gram models without first clustering the dataset. The algorithm is loosely inspired by the growing gas algorithm [44].

The novel Growing N-Gram (GNG) algorithm samples strings from the input corpus and identifies nearest N-Gram models. Depending on the set difference between the string and existing models a new model is created, or the nearest model is updated. The set difference between a string W and model M is defined as the number of unique symbols in W that are not in the unique symbols of the training corpus composing M. The lower the set difference, the string to that model and vice versa.

Strings that are not close enough to models to update them and not far away enough to be certain a new model needs to be created are skipped for the next update/create iteration. The create/update/skip actions are determined by two thresholds. These two thresholds ensure within model similarity while maximizing between model distance. As the algorithm evolves, models get updated and new models are created. It might be that a newly created model better fits the skipped strings, or maybe one of the existing models is updated such that it better fits the string. The graphical representation of the GNG algorithm is presented in Figure 13 .



**Figure 13: Graphical representation of the GNG Algorithm. Green dots represent strings, orange squares represent N-Gram models, the grey area represents the "wait" region**

5.2.4 Probabilistic Usage Profiling Validation

Independently on the algorithmic approach taken for clustering, the result of our usage modelling corresponds to a set of k N-Gram models, where k is the number of clusters partitioning the input corpus. We refer to the set of k N-Grams as an ensemble of models, or simply an ensemble.

One problem related to unsupervised learning resides on methods for validating the resulting clusters: how can we understand whether the partitioning of the corpus of usage strings into k clusters performed by algorithm A is *better* than the partitioning into w clusters performed by algorithm B?

**Quantitative Validation**

Clearly, several measures for validation exist, though many of them cannot be directly used in our string-based use case: we have seen that string similarity measures are problematic and diverse. Another problem resides on the fact that using e.g. string-based similarity measure A to compare the clustering results obtained when string similarities A and B are taken into account seems unbalanced. Moreover, we do not have any indication on whether a partitioning into few big clusters would be more ideal than one into many small clusters or a mixed situation composed of few big and many small clusters (or vice versa). We cannot even directly rely on domain expert knowledge, since there is no consensus on how many behaviours should exist and how they would look like.

In order to tackle this problem, we made a fundamental statement: our goal is not partitioning the training corpus into clusters, rather being able to classify (unseen) strings to the best of our capabilities.

In other words, we must be able to formulate statements such as “input string y belongs to cluster k with 99.99% confidence” rather than “input string y might belong to either cluster k (33% confidence), cluster l (34% confidence) or cluster m (32% confidence).

Our statement leverages on the following considerations:

* Our usage models have a probabilistic nature. The calculation of a classification confidence can not only be retrieved via chain probabilities, but can also indirectly provide a feedback on whether two or more clusters overlap (their resulting chain probabilities would be very similar)
* The confidence calculation is comparable across approaches which would return a different number of clusters

Figure 14 (a) represents the ideal situation with one winning model, whereas the chain probabilities in Figure 14 (c) are more closely related to a uniform distribution. A uniform distribution or close to uniform means that multiple models are equally likely to represent some string, suggesting an overlap of strings in corpora of different models. Since clustering aims to minimize within cluster distance while maximizing between cluster distance, corpora are not supposed to contain similar strings. A uniform distribution or close to uniform is therefore an indication of poor clustering.

The closer the probability distribution is to the ideal situation, the better the ensemble represents specific clusters of strings. Since the probability distribution in Figure 14 (b) is more related to the winning ideal situation, whereas Figure 14 (c) is more related to a uniform distribution, Figure 14 (b) is preferred over (c).

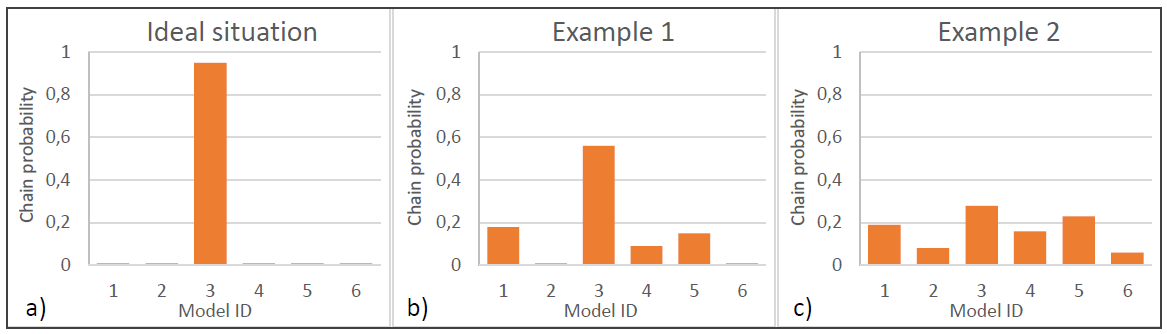


Figure 14: Usage classification scenarios. (a) ideal situation, (b) and (c) average situations

To achieve the evaluation of our probabilistic classifier performances we analyse the ratio of the highest chain probability of an ensemble over the uniform probability of that ensemble. Such *deviation from uniform distribution* allows us to be able to compare classification performance across ensembles of different sizes. For instance, with respect to Figure 14 (b), the deviation from uniform distribution would be calculated as the ratio of the highest score recorded by the third model divided by the uniform distribution 1/6, where 6 is the number of models composing the ensemble.

**Crowdsourced Validation**

The quantitative evaluation of the resulting clusters and therefore usage models can give an indication on how well the used algorithm partition the training corpus and subsequently classify unseen usage examples. Nevertheless, we should still rely on expert knowledge as a form of validation.

As previously stated, some of the drawbacks existing when domain expert knowledge is used are:

* experts can be knowledgeful on a very specific aspect of machine usage;
* experts are biased towards how they think machines should be operated;
* other experts can have different opinions on the same topics;
* experts, when observing usage model graphs composed of many nodes and edges, might encounter difficulties in analysing them;

An intuitive way to overcome or at least reduce these issues is by relying on as many experts as possible. We achieved so by organising an online crowdsourced experiment, involving participants with different levels of expertise and professional areas (e.g. radiologists and system testers). Moreover, we asked them to evaluate usage instances rather than N-gram models. Our assumption is that, in case our models well describe different usages, a random walk through the probabilistic graphs would lead to a *believable* usage instance.

This form of indirect validation – experts do not examine the models, rather the traces resulting from randomly walking through them – is also highly suitable when the usage models are black boxes or highly complex. For instance, when we consider the use of N-grams with N > 2, experts would have to evaluate a large amount of conditional probability tables rather than graphs. However, since we perform random walks on graphs, the possible paths taken are virtually infinite, the exhaustive validation of even a single model remains problematic and left for future investigations.

Our crowdsourcing experiment is very similar to the famous Turing test for artificial intelligence [45]. In a nutshell, we created a dataset composed of *real* bolus-chase procedures – i.e. procedures retrieved straight from logs – and *synthetic* procedures – i.e. the traces obtained by randomly walking through the usage models. We then asked a simple yes/no question to the experts: “*does this sequence of actions occur in clinical practice during Bolus Chase?*”. Additional optional feedback can be provided by the participants, such as highlighting specific unexpected sub-sequences or tagging full sequences with clinical-specific terms.

Our aim is that the ratio of “yes” answers for synthetic procedures would approach the same ratio for real procedures, which would allow us to conclude that our usage models successfully mimic the behaviours found in real life, at least for the amount of synthetic procedures generated. The experiment was launched on 10th April 2018 and data will be collected at least until the end of Summer 2018. A screenshot of the Turing Test is presented in Figure 15.

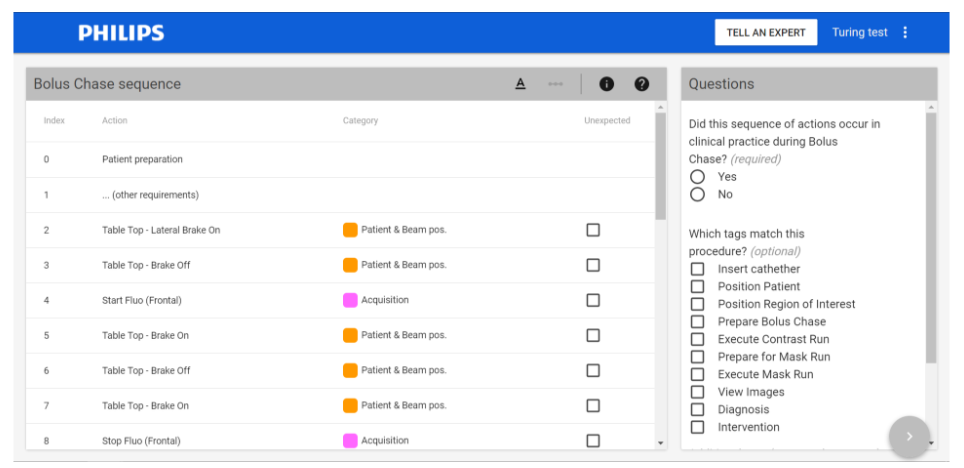


Figure 15: Screenshot of the Turing test for expert-driven usage modelling validation

5.3 Hierarchical Usage Profiling

Another approach to process the log information for usage profiling is not to leverage on the sequence of actions, but rather on the statistical distribution of these throughout examination and on the hierarchical structure existing during a medical procedure.

The work described in this Section was conducted with the clear intention to devise models for test case generation, though the generalisation of the approach beyond test case generation can be trivially imagined. Therefore, in order to define the statistics needed for usage profiling, it is important to know how a test case looks like. With respect to Figure 8, we can observe a recurring theme in all patient treatments: they always start with preparation, a set of clinical procedures are executed, and they finish off with the completion phase. A test case should do exactly that, simulate treatment of a patient.

Preparation and completion phases are fixed among all test cases. Therefore, it is not necessary to include statistics for them in the user profile. They are modelled once and reused in every test case. As for the examination phase, the generated test case should include at least one clinical procedure. Each procedure is defined in terms of the actions that can be executed on the system. Therefore, apart from statistics at the procedure level, we also need statistics at the level of actions. We need to know the frequency of actions per procedure and also the distribution of the data values for each action. There is another level of statistics above the other two and that is the patient type (denoted as procedure combination in Figure 16). Clinical procedures are grouped to denote the type of the patient.

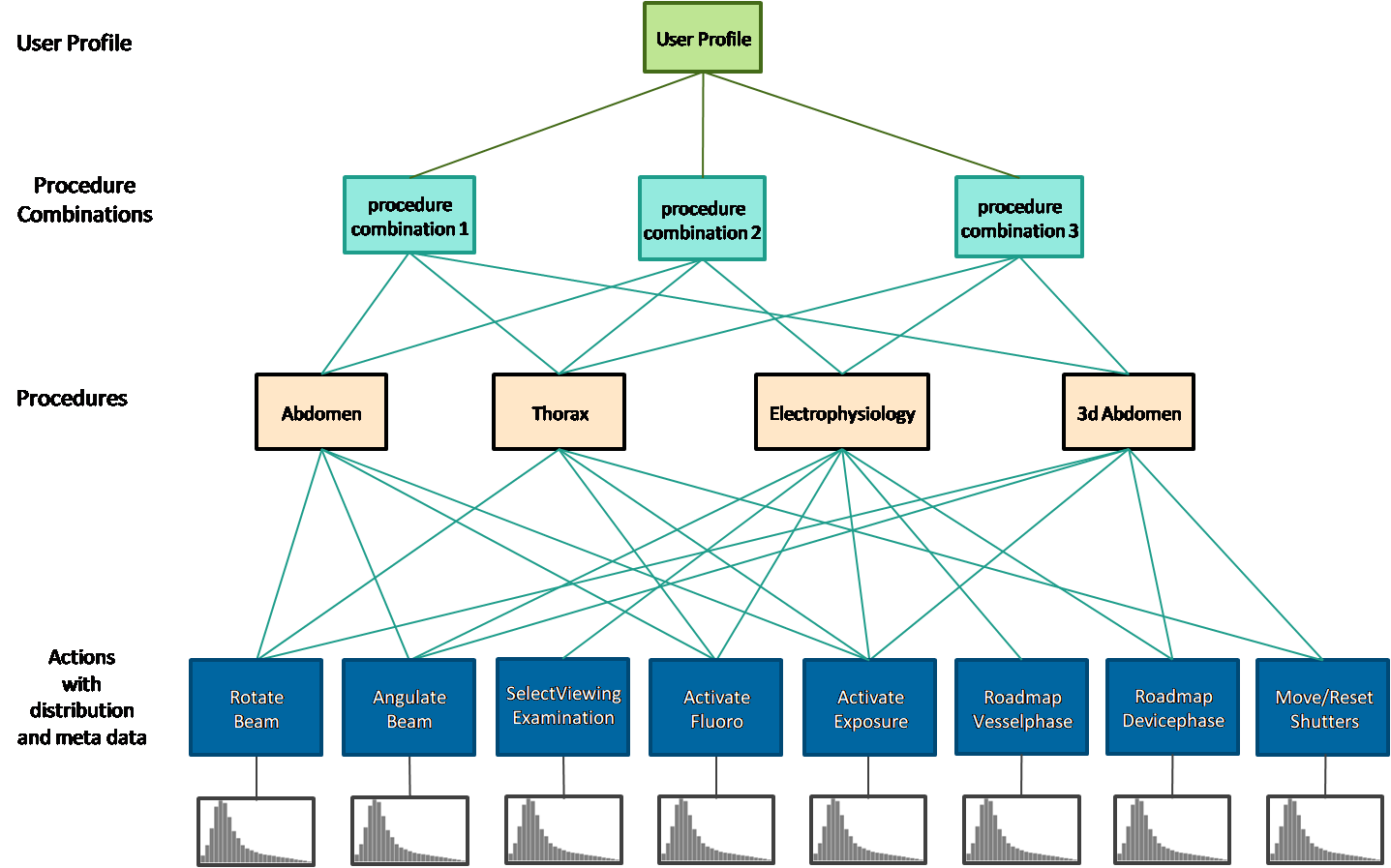


Figure 16: Structure of user profile

Percentages are used to report the statistics at the first two levels. For example, we could have a profile in which 30% of the field usage belongs to Cardio patients and 70% belongs to Neuro. One level deeper, a procedure cluster like Cardio can consist of several clinical procedures each with a percentage of its own occurrence in the log data.

At the lowest level we have the mapping of procedures to their actions. Two distributions are reported per action:

* A concrete distribution showing the frequency of the action per clinical procedure (Figure 17). The x-axis in the picture represents the number of times the action was performed, and the y-axis represents the number of clinical procedures (all of the same procedure type). Suppose Figure 17is the frequency distribution for Beam Angulation of a procedure like Thorax. Then the yellow vertical bar means there were 89 Thorax procedures that angulated the beam 6 times.
* An abstract distribution consisting of the minimum, maximum, average and standard deviation of the data values with which the actions was performed on the system.

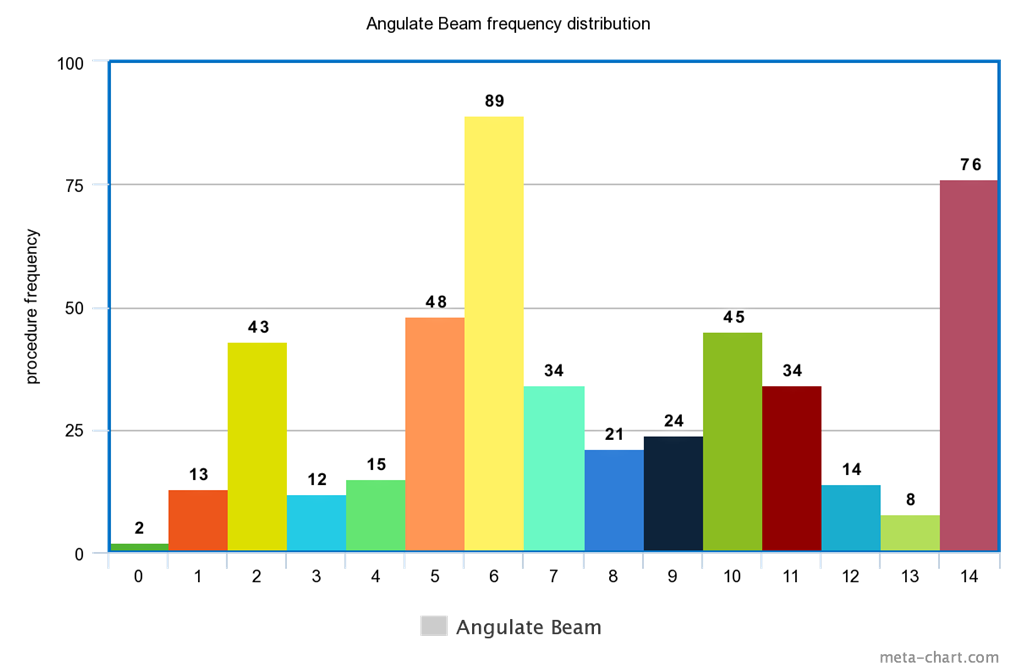


Figure 17: Action frequencies

### 5.4 Usage Profiling for System Verification / Reliability Testing

Every release, the x-ray systems are thoroughly tested to ensure it satisfies all technical, medical and safety criteria. Besides this, the system reliability is also verified. The goal of reliability testing is to drive a system with realistic clinical scenarios in conditions which were not expected beforehand, taking care that the system is not load and stress tested. Currently, a few log files which were converted into automatic test scripts are used over and over, running for a few thousand hours per system release. As time goes by, these scripts have been adapted to make sure they keep running, resulting in scripts that barely resemble a realistic clinical scenario. Therefore, a new and robust method is needed which provides a practically endless source of automatic test scripts.

5.4.1 classify-and-test approach via probabilistic usage profiling

Usage models, being generalisations of similar behaviours, are imperfect. For instance, there is always a non-zero chance that a random walk would lead to a sequence of actions which does not occur in clinical practice during Bolus Chase. This means, with respect to the Turing test described in Section 5.2.9, that even if all synthetic procedures generated by model X scored 100% “yes”, that does not prove that the same model will always generate “yes”-type procedures.

This also means that, should we want to attempt the use of probabilistic models for automated test case generation, further investigation on how to reduce or even remove invalid sequences is required. Nevertheless, probabilistic usage models, which also function as usage classifiers, could be used for system reliability verification testing as an aid for domain experts.

Imagine a scenario in which system verification tests are manually performed by executing in-field machine logs. Experts have the daunting challenge of selecting the most appropriate logs to be tested, depending on what kind of tests they wish to perform. A help in pre-selecting the logs, or at least in suggesting which logs should be tested first, could come from usage model classification. By using usage model classification, we can propose which log files are considered as normal, odd or even exceptional behaviour. For example, the log file generated on a demo or training day should not be included for reliability verification.

The K models resulting from the clustering approach – or GNG algorithm – although are capable of classifying input logs, are not perfect in labelling logs as normal / odd / exceptional behaviours. Our solution is the implementation of a feedback loop, from expert to models, such that the labels would come from the expert, and in this way the second classification iteration would leverage on the labels obtained at the previous loops. The following Figure depicts the schematics of the mixed-initiative classification task.



Figure 18 : Mixed-initiative system verification testing

By creating such cooperative environment, the modelling/classification task would improve its classification performance based on expert knowledge, and the expert would certainly benefit from the classification ranking coming from the modelling/classification ensemble, greatly downscaling the manual analysis of the input logs.

Extension of this work is straightforward: once the expert is satisfied enough of the classification task, the expert could toggle herself from the loop, therefore approaching a form of automated and endless testing. Furthermore, being the ensembles learning both from data and from the experts, we could even achieve the possibility to model, at the same time, usage behaviours and different expert beliefs – i.e. different ensembles which cooperated with different experts would provide different classification results.

5.4.2 Test Script Generation via Hierarchical Usage Profiling

This approach is centred on Axini’s model-based testing tool (Modelling Suite) [46] to make usage profile-based testing operational with the clear objective to lift testing from a script-based approach - see Figure 19 and Section 5.3 – to complete automation. Axini’s tool can automatically generate test cases - TAF scripts in Figure 19 – that reflect the usage of the system in the field. These test cases hit the functionalities of the system according to their frequency of use.

This approach, similarly to the previous one of Section 5.5.1, uses the log files from the field as input for test case generation. Philips’s Qlikview application [47] provides an interface to query over the log files based on different criteria. Once a subset of the log data is selected, a user profile can be generated from it. Axini’s Modelling Suite uses the usage profile in combination with a model and a testing strategy to generate test cases.

The key-contributions of this approach include:

* defining the format of the user profile tailored for Philip’s use case;
* extending the Axini Modeling Suite to accommodate user profile testing;
* generating test cases in a format that can be executed on the system under test.

**Usage Profile-driven Model-based Testing**

This research makes usage profile testing operational by combining it with model-based testing. Manually testing complex systems is both hard, insufficient and not scalable. Model-based testing is a way to tackle those issues. The amount of test cases generated from a model can be infinite. The only requirement is a test criteria to guide the selection of test cases. Usage profiles can be used as a test selection method, as they specify the usage of the system in production. Their combination with the model steers testing towards generating test cases that represent a real life use of the system in the field. This allows to validate and test areas of the system that are known to be used more often.

Figure 21 shows the integration of the usage profile of Section 5.3 with Axini’s way of model-based testing. The key ingredients are the model, the testing strategies and the usage profile. The model specifies the actions that can be executed on the system. The usage profile is a statistical distribution over the actions in the model. The testing strategy uses both the model and the usage profile to generate test cases according to the statistical properties extracted from the log files.

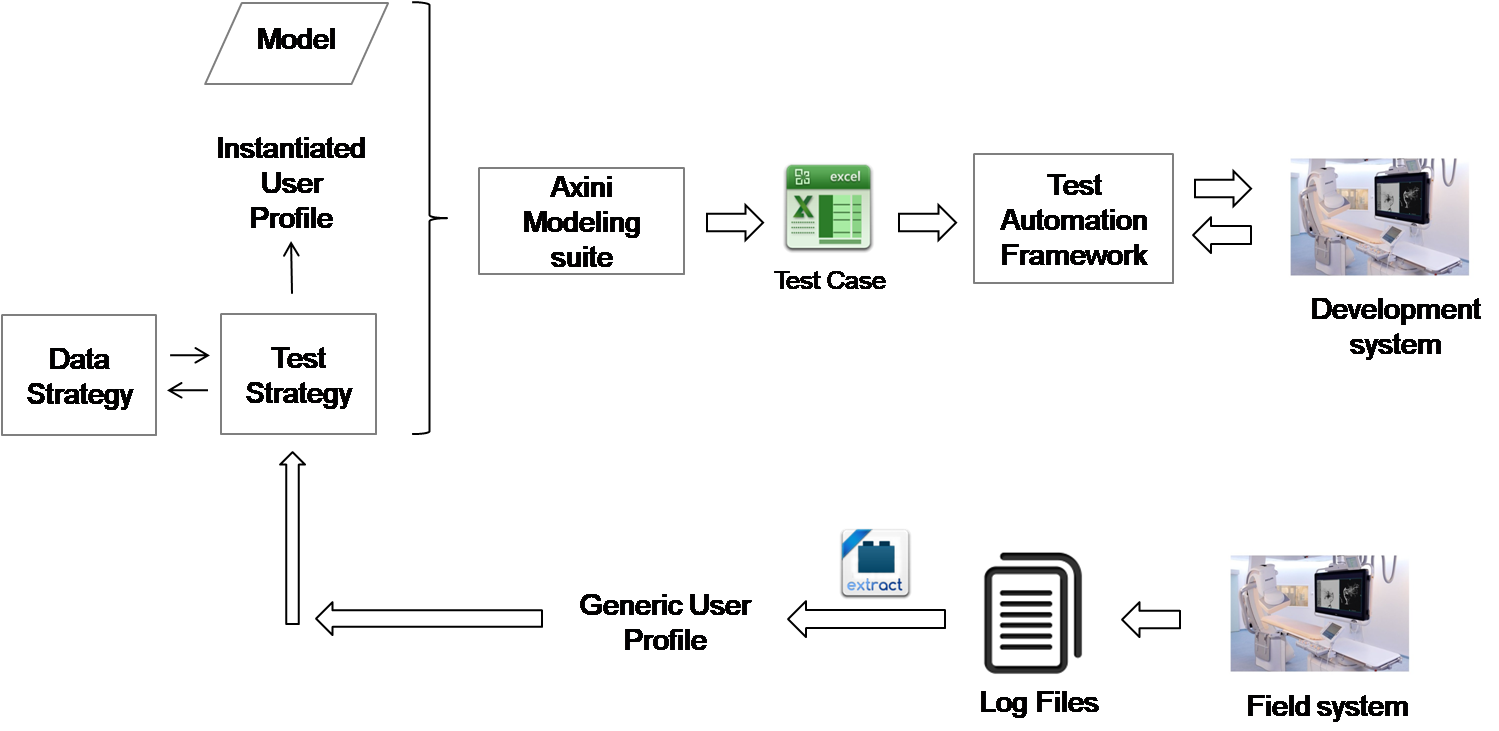


Figure 19: Usage Profile-driven Model-based Testing

The usage profile contains statistics at several levels: patient-type, clinical procedures, actions and the corresponding data values. Each level has a distribution of its own. The testing strategy takes this generic data for every test case. The result of this instantiation is a probability distribution over the actions in the model which can also be viewed as weights that are dynamically attached to the transitions in the model. By keeping track of the frequency of actions that have been covered so far, the testing strategy is able to generate test suites that target the distributions in the user profile.

Apart from behaviour, the data attributes with which the behaviour is triggered is also present in the usage profile, therefore they are reflected in the generated tests. The data strategy has knowledge of it and is consulted by the test strategy every time an action is to be instantiated.

**Test Case Generation**

This section explains, at a high level, how test cases are generated. Recalling the workflow from Figure 21, several components are at play: model, user profile, testing strategy and the data strategy. Figure 20 shows an example model with four actions that can be executed on the system. It only focuses on the examination phase and abstracts from preparation and completion. At least one clinical procedure should be executed in a test case, so the model starts by choosing a procedure combination. The testing strategy has knowledge of the usage profile, since it has already parsed it before the test generation started. By taking the statistics at the first level into account, the test strategy picks a procedure combination. The next step corresponds to choosing a procedure. The test strategy does that in a similar way, this time by using the statistics at the second level that are scoped to the selected procedure combination. Once a procedure is defined, the usage profile specifies which actions are possible and which ones are not. The model specifies everything that can be done on the system while the strategy filters the actions based on the data from the user profile during test generation.

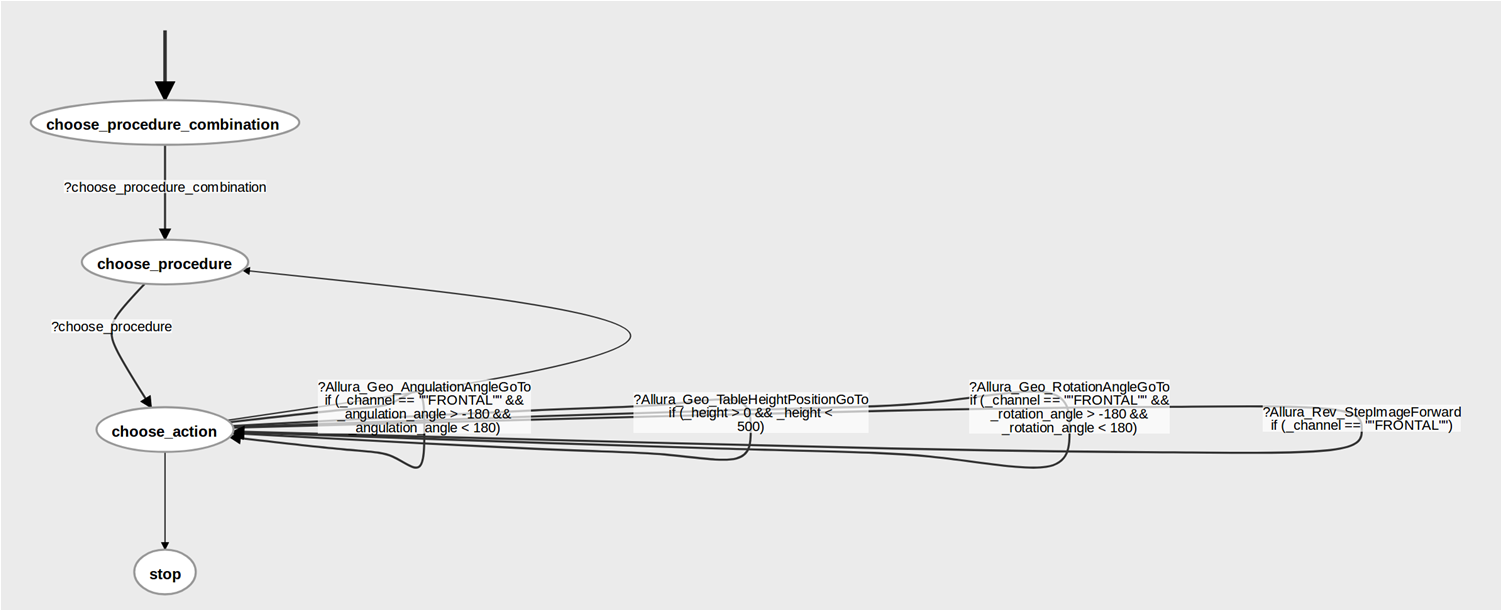


Figure 20: Model Example

Each action has a concrete frequency distribution. Using the weighted random sampling algorithm [48], the test strategy calculates a probability for each action and attaches it to the corresponding transition in the model (see Figure 21).

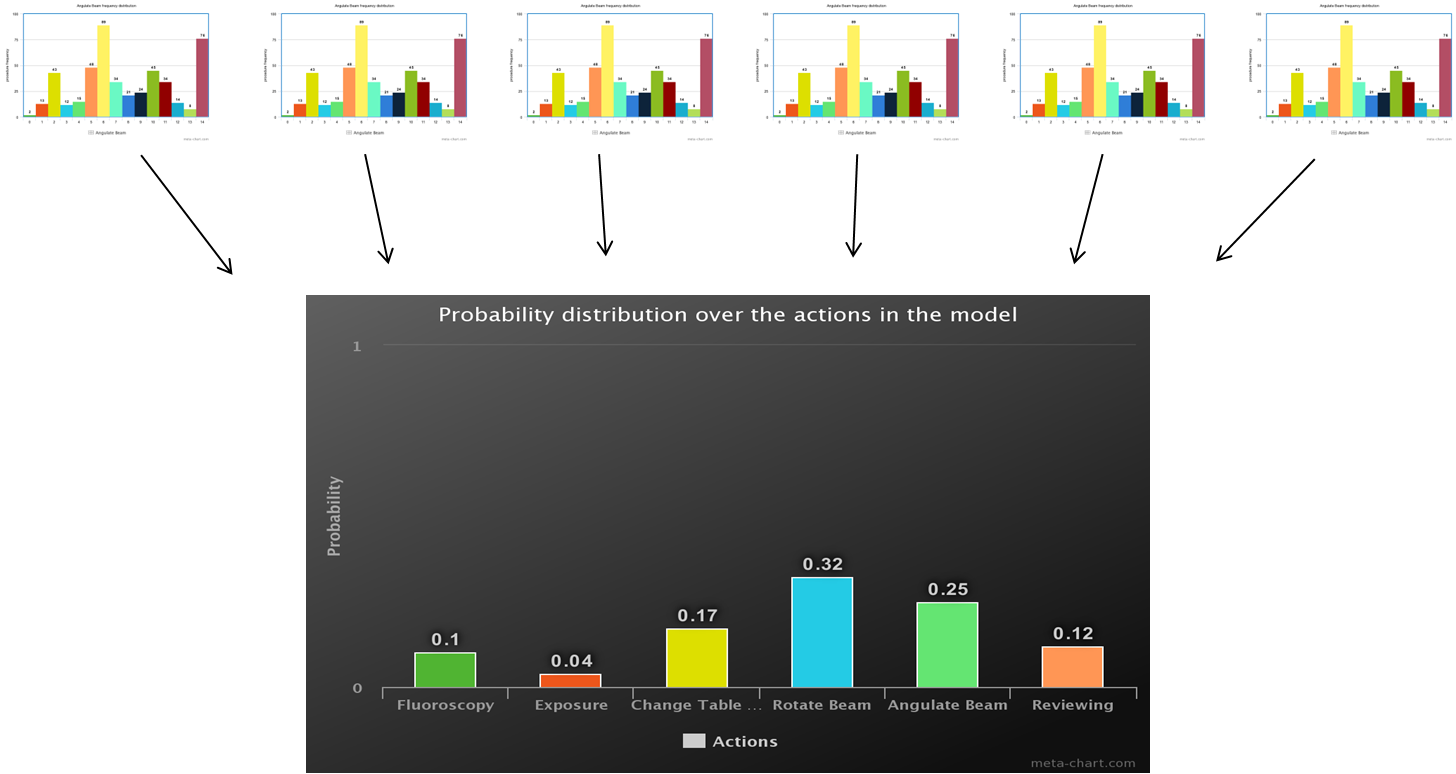


Figure 21 Weighted Random Sampling

This top-to-bottom instantiation of the usage profile happens with every test case that is generated from the model. The test strategy also maintains a history of the actions, procedures and procedure combinations that have been used so far in the test suite and takes that into account in its instantiation algorithm. This way the strategy keeps track of how much of the user profile has been targeted by the test suite.

Finally, the data strategy that is used whenever an action is chosen by the test strategy is retrieved. The data strategy completes the instantiation by picking a value for the action. The data strategy can be configured with one of the following selection strategies (see Figure 22):

* normal use: most common used functionality and intensity. One standard deviation away from the mean: µ ± σ
* alternative use: average functional and intensity. Within the σ and 2 σ standard deviation areas;
* exceptional use: functional corner cases and intensity limits. Exceeding the 2 σ standard deviation areas.

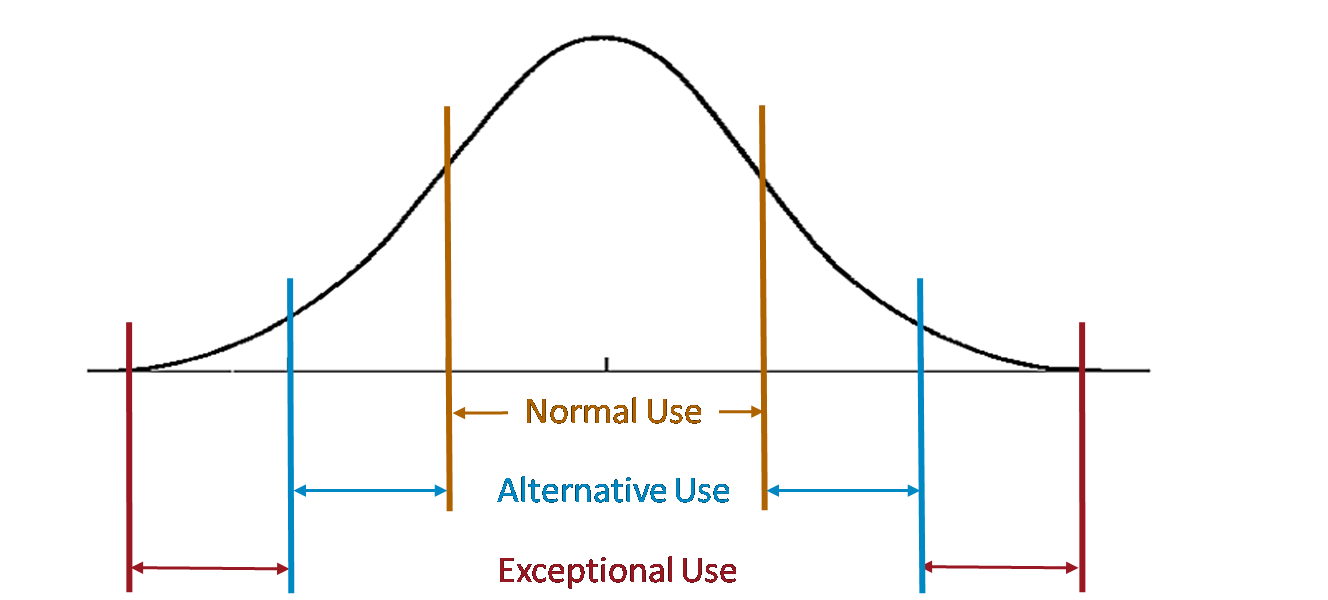


Figure 22 Data Strategy Configurations

1. Case study: Usage profiles for system requirements in the context of Philips MR

Understanding how the customer uses the system and how its behaviour deviates from the expected (and designed) one is the main question that Philips MR wanted to answer by using usage profiling. Philips MR is a division of Philips Healthcare that builds, as the name suggests, systems for magnetic resonance imaging (MRI). MRI is a non-invasive diagnostic imaging method.



Figure 23: Philips MRI Scanner

An MR system (see Figure 23) can be substantially parametrized due to the fact that there are quite a lot of new development in the field with many new methods appearing constantly. For this reason, guidelines for usage with respect to a particular application (diagnostic) are most of the time vague. Therefore, usage profiling for an MR system starts with answering how one can define usage.

We present in the first two subsections a description of the data and how the data is used to define the usage. In the second subsection, we focus on the use of process mining to create usage profiles and how these profiles can be used in combination with medical guidelines.

1. Data overview

An MRI examination is defined by its purpose (the diagnostic part) and the method(s) applied. We choose to abstract the purpose in terms of the anatomic region that needs to be imaged. In terms of the method, practitioners use a set of scans to produce multiple images that will later on provide evidence for/against a particular diagnosis. Each scan is in fact defined by the parameters used to set-up the machine. Figure 24 presents a summary of the data structure:

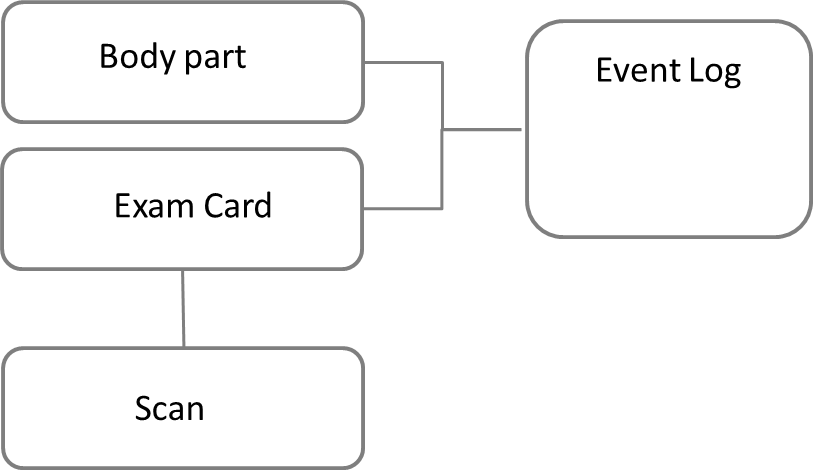


Figure 24: Main data items for an MRI scanner

One should note that the set of parameters available for a scan depend on the characteristics of a particular system. Therefore, we decided early on to focus our investigation on a particular system release.

1. Defining the usage

As mentioned in the previous subsection, the usage of an MRI systems is defined by the performed examinations. At the lowest level, the usage is thus represented by the parameters of a scan. However, when trying to use all parameters used for a scan to define a scan we realize that comparing two scans becomes a highly complex task for two reasons: 1) for a specific scan, in average, less than 10% of the parameters are used and, 2), the parameters types are highly heterogeneous: categorical, numerical and Boolean.

An additional challenge was the fact that since the data in our possess was created for debugging purposes, the granularity and the definitions of the parameters are different than the ones expected by practitioners.

A solution to the above challenges was proposed by mapping the logged parameters to so-called “tags” defined by MRI literature and, at the same time, selecting a reduced number of tags to represent a scan. For the mapping and selection, we used input from medical guidelines and practitioners.

The proposed approach made scan parameters easily understandable by practitioners and facilitated exam analysis based on expected behaviour and medical guidelines.

1. From scan parameters to profiles

Figure 25 presents the implemented workflow to define and use profiles. First, as mentioned in the previous subsection, we defined a mapping from actual scan parameters to “tags”. We use domain-specific language (DSL) technology (represented by combination of Xtext/Xtend) to allow Philips specialists to define the mapping. Once such a mapping is created the framework automatically generates python code that tags the extracted data.

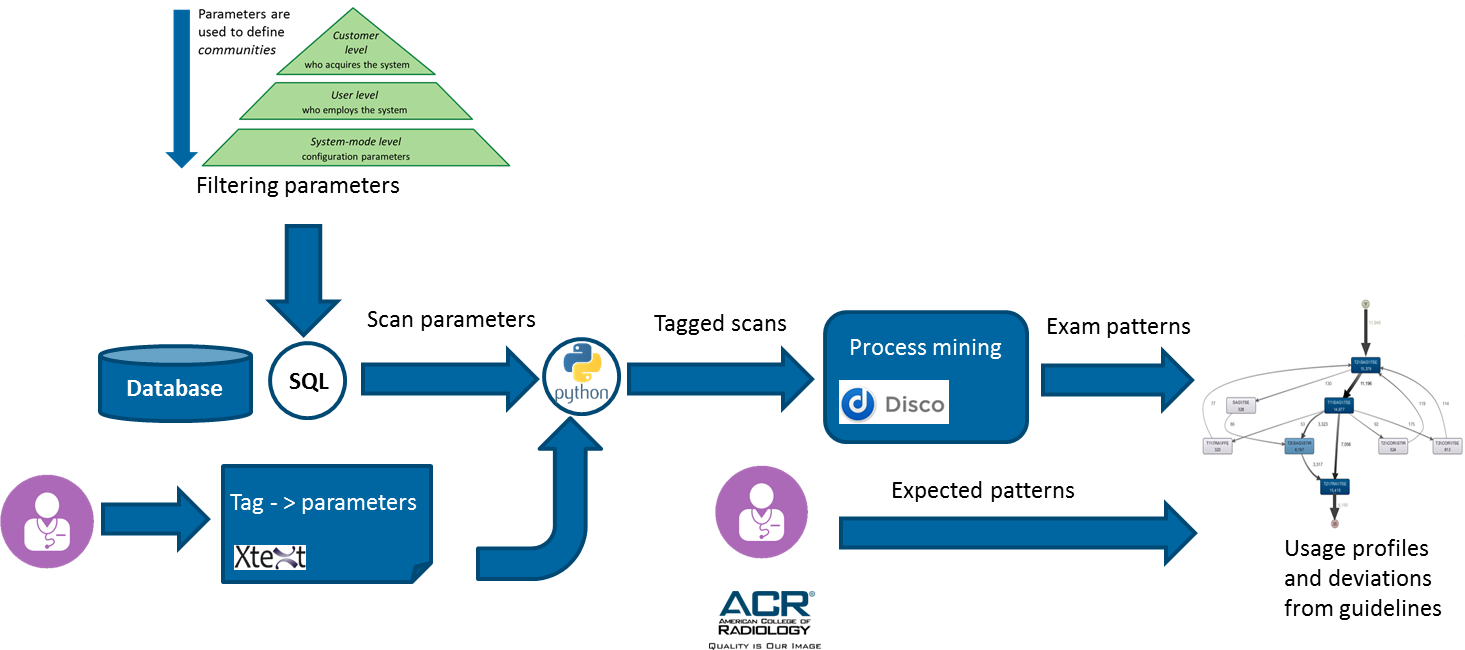


Figure 25: Processing workflow for creating usage profiles

The usage profiles created are group profiles of type community (as defined in Section 2.1) where each community is defined by criteria from customer level (e.g. region), user level (e.g. body part) and system-mode level (e.g. 1.5T system). For each community such defined we extract available data from logs database that we further map to tags as described above. Table 1 presents an excerpt of the data log after tagging for a community defined by region EMEA, system mode 1.5 and body part Lumbar spine.

Table 1: Log excerpt for an MRI scanner

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Case id | Scan  parameters | Timestamp | Region | System  mode | Body part |
| 0 | T2\_SAG\_TSE | 13:50.4 | EMEA | 1.5 | Lumbar spine |
| 0 | T1\_SAG\_TSE | 17:44.9 | EMEA | 1.5 | Lumbar spine |
| 0 | T1\_SAG\_TSE | 22:41.6 | EMEA | 1.5 | Lumbar spine |
| 0 | T2\_TRA\_TSE | 26:34.5 | EMEA | 1.5 | Lumbar spine |
| 1 | T2\_SAG\_TSE | 07:51.3 | EMEA | 1.5 | Lumbar spine |
| 1 | T1\_SAG\_TSE | 12:46.5 | EMEA | 1.5 | Lumbar spine |
| 1 | T1\_SAG\_TSE | 18:01.4 | EMEA | 1.5 | Lumbar spine |
| 1 | T2\_TRA\_TSE | 23:14.4 | EMEA | 1.5 | Lumbar spine |
| 3 | T2\_SAG\_TSE | 37:12.7 | EMEA | 1.5 | Lumbar spine |
| 3 | T1\_SAG\_TSE | 40:54.5 | EMEA | 1.5 | Lumbar spine |
| 3 | T2\_SAG\_STIR | 44:16.7 | EMEA | 1.5 | Lumbar spine |
| 3 | T1\_SAG\_TSE | 48:13.7 | EMEA | 1.5 | Lumbar spine |
| 3 | T1\_TRA\_FFE | 53:17.6 | EMEA | 1.5 | Lumbar spine |
| 5 | T2\_SAG\_TSE | 13:43.1 | EMEA | 1.5 | Lumbar spine |
| 5 | T1\_SAG\_TSE | 15:40.5 | EMEA | 1.5 | Lumbar spine |
| 5 | T1\_TRA\_FFE | 18:54.3 | EMEA | 1.5 | Lumbar spine |
| 5 | T1\_SAG\_TSE | 21:24.4 | EMEA | 1.5 | Lumbar spine |
| 5 | T2\_TRA\_TSE | 22:56.3 | EMEA | 1.5 | Lumbar spine |

For each such obtained input data, we obtain a workflow representation by means process mining techniques. Process mining is defined as the field of data science that extracts directed graph-type models where, in most cases, each node represents a performed activity (in our case a scan) and the edge connecting two nodes represents the flow of work (i.e. following scan to be executed).

|  |  |
| --- | --- |
| Usage profile through process mining | Excerpt from “ACR–ASNR–SCBT-MR practice parameter for the performance of magnetic resonance imaging (MRI) of the adult spine” (<https://www.acr.org/-/media/ACR/Files/Practice-Parameters/MR-Adult-Spine.pdf> ) |

Figure 26: Side-by-side a usage profile and a medical guideline excerpt

Once the usage profile is obtained, a practitioner can compare the workflow with known medical guidelines (such as the ones provided by American College of Radiology – ACR). In Figure 26 we present side by side a usage profile and an excerpt from the guidelines. Note that the thickness of the edge is correlated to the number of direct relations between the scans: the thicker the edge, the more frequently the relation is observed in the data. It is easy to observe that most typical workflow is the one indicated in the guidelines: T1 Sagittal => T2 Sagittal => T2 Transversal (or Axial). However, a number of deviations are observed. These deviations are currently investigated by practitioners to understand whether there are special workflows employed by certain practitioners or there are anomalies due to system/user error.

1. Conclusions

This report presented an overview of the profiling process highlighting its particularities with respect to knowledge discovery process. We based our remarks on literature reviews and on the industrial use cases.

When profiling, we emphasize the importance of understanding what profiles are relevant to our purpose and where the information can be found. After gathering the relevant data, we identify the particularities of knowledge discovery process for profiling. We emphasize that one major issue is translating available data into relevant information for the defined objectives. Additionally, we focus on two important aspects of discovery: model discovery and model fitting. In model discovery, data is the first-class citizen determining the relations between the important features, where in model fitting, the data is used to refine or add information to a-priori defined profiles. The two aspects are complementary.

We focussed on two particular applications for profiling based on the current use cases provided by our industrial partners: requirements engineering and system reliability. In the first case, we note that by using available information from the installed systems, system architects and business developers can not only validate their assumptions with respect to the current usage, but also understand which functionality is critical at this moment and where improvements are needed. Moreover, data can lead to customer profiles associated with certain usage patterns that can improve, e.g., marketing strategies. For the latter, usage profiles provide a good overview for validation engineers to understand the occurrence rate of the system functions. At the same time, data provides insight into the workflow employed by customers when operating the systems.

We first identified a use case study – namely the interventional x-ray machines of Philips Healthcare’s Image Guided Therapy (IGT) – and we approached the usage profiling task in two ways. The firs approach relied on several machine learning techniques aimed at partitioning the dataset into clusters so that within cluster similarity is maximised and between clustering similarity is minimised. Subsequently, each cluster was modelled via N-Grams, ultimately leading to ensembles of probabilistic models, one model for each identified usage behaviour. The research not only relied on well-established and new techniques for clustering, but also proposed a novel algorithm, called Growing N-Grams, which attempts to solve the issues related to the definition of string similarity measures. The second approach, on the other hand, relied on the existing hierarchical structure of the medical procedures performed with IGT machines, to extract detailed statistical information of each action composing such procedures.

We also investigated what validation techniques could be used, albeit solely for the probabilistic approach to usage modelling. We identified a quantitative approach aimed at determining which ensembles results to a better classification task, implying that the clustering task successfully partitions the input corpus into non-overlapping behaviours. We also attempted a qualitative validation approach by relying on domain expert knowledge. We organised an online crowdsourced experiment where experts would analyse sequences – either synthetic or real-life procedures – and subsequently evaluate whether those sequences would be observed in practice.

Our research was conducted with the idea to eventually improve system verification testing. The two modelling approaches were used in different ways: the probabilistic approach was embedded in a cooperative loop where the other actor is the expert/tester. The result is a mixed-initiative adaptive system, where machine learning performs an initial modelling and classification of input behaviours, followed by a final validation of such classification performed by the expert. A feedback loop would then be used by the classifier to adjust its classification rationale. The hierarchical approach to usage modelling, instead, was embedded in Axini’s Modelling Suite tool in order to generate usage-based and model-based test script. Both work showed that usage modelling can be used in ways which would allow the transitioning from manual to fully automated and endless testing.

We also attempted the use of process mining and domain-specific languages for another case study – the extraction of usage profiles of magnetic resonance imagining systems of Philips Healthcare’s MR. The main focus of this research was to understand how and whether these systems adhere to official medical guidelines.

As it can be understood, usage profiling is a task which can be performed in many several ways – e.g. via machine learning, process mining or statistical analysis, to name few. It is therefore important to clearly outline what should be modelled and for which purposes, since it might happen that some approaches might become more suitable than others. Nevertheless, and possibly more importantly, the possibility to adopt several data-driven techniques (and expert-driven ones, e.g. domain-specific languages) concurrently allows to further investigate how these could be used cooperatively rather than exclusively, so that the strengths of one approach can overcome the weaknesses of another.

# Bibliography

|  |  |
| --- | --- |
| [1] | M. Hildebrandt and S. Gutwirth, Profiling the European Citizen: Cross-Disciplinary Perspectives, Springer Publishing Company, Incorporated, 2008. |
| [2] | S. Pachidi , M. Spruit and I. van der Weerd, "Understanding users’ behavior with software operation data mining," *Computers in Human Behavior,* vol. 30, pp. 583-594, 2014. |
| [3] | C. Smidts, C. Mutha, M. Rodríguez and M. J. Gerber, "Software testing with an operational profile: Op definition," *ACM Computing Surveys (CSUR),* vol. 46, no. 3, p. 39, 2014. |
| [4] | M. K. a. J. P. Jiawei Han, Data Mining: Concepts and Techniques (Third Edition), Morgan Kaufmann Publishers, 2011. |
| [5] | C. Gunther, A. Rozinat, W. van der Aalst and K. van Uden, "Monitoring Deployed Application Usage with Process Mining," BPM Center Report BPM-08-11, BPMcenter.org, 2008., 2008. |
| [6] | C. Ghezzi, M. Pezze, M. Sama and G. Tamburrelli, "Mining Behavior Models from User-intensive Web Applications," in *Proceedings of the 36th International Conference on Software Engineering*, 2014. |
| [7] | B. Mobasher, "Data Mining for Web Personalization," in *The Adaptive Web: Methods and Strategies of Web Personalization*, Springer Berlin Heidelberg, 2007. |
| [8] | M. El-Ramly and E. Stroulia, "Mining software usage data," in *Proceedings of 1st International Workshop on Mining Software Repositories* , 2004. |
| [9] | M. Andriy, A. Hamou-Lhadj, E. Cialini and A. Larsson, "Operational-Log Analysis for Big Data Systems: Challenges and Solutions," *IEEE Software,* vol. 33, no. 2, pp. 52-59, 2016. |
| [10] | A. Oliner, A. Ganapathi and W. Xu , "Advances and Challenges in Log Analysis," *Communications of the ACM,* vol. 55, no. 2, pp. 55-61, 2012. |
| [11] | S. Alspaugh, A. Ganapathi, M. A. Hearst and R. Katz, "Better Logging to Improve Interactive Data Analysis Tools," in *Workshop on Interactive Data Exploration and Analytics (IDEA). Co-located with KDD*, 2014. |
| [12] | U. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *Communications of the ACM,* vol. 39, no. 11, pp. 27-34, 1996. |
| [13] | L. Kurgan and P. Musilek, "A Survey of Knowledge Discovery and Data Mining Process Models," *The Knowledge Engineering Review,* vol. 21, no. 1, pp. 1-24, 2006. |
| [14] | R. J. C. Bose and W. van der Aalst, "Abstractions in process mining: A taxonomy of patterns," *International Conference on Business Process Management Proceedings,* pp. 159-175, 2009. |
| [15] | J. C. T. M. R Michalski, Machine learning: An artificial intelligence approach, Springer Science & Business Media, 2013. |
| [16] | J. P. J. M. K. J Han, Data mining: concepts and techniques, Elsevier, 2011. |
| [17] | W. van der Aalst, Process Mining, Discovery, Conformance and Enhancement of Business Processes, Springer-Verlag Berlin Heidelberg, 2011. |
| [18] | R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *Proceedings of the ACM SIGMOD International Conference on Management of Data ,* vol. 22, no. 2, pp. 207-216, 1993. |
| [19] | N. Tax, N. Sidorova, R. Haakma and W. v. d. Aalst, "Event Abstraction for Process Mining using Supervised Learning Techniques," in *Proceedings of the SAI Intelligent Systems Conference*, London, 2016. |
| [20] | H. Schonenberg, B. Weber, B. v. Dongenl and W. v. d. Aalst, "Supporting Flexible Processes through Recommendations Based on History," *Lecture Notes in Computer Science,* vol. 5240 , pp. 51-66, 2008. |
| [21] | A. M. Davis, "The art of requirements triage," *Computer,* vol. 36, no. 3, pp. 42-49, 2003. |
| [22] | W. Maalej, M. Nayebi, T. Johann and G. Ruhe, "Toward Data-Driven Requirements Engineering," *IEEE Software ,* vol. 33, no. 1, pp. 48 - 54, 2016. |
| [23] | R. Gorrieri, A. Rensink and M. A. Zamboni, "Action Refinement," in *Handbook of Proacess Algebra*, Elsevier, 2000, pp. 1047-1147. |
| [24] | D. Jurafsky, Speech and language processing: An introduction to natural language processing., Computational linguistics, and speech recognition, 2000. |
| [25] | S. F. &. G. J. Chen, "An empirical study of smoothing techniques for language modeling.," *Computer Speech & Language,* vol. 13, no. 4, pp. 359-394, 1999. |
| [26] | D. P. a. J. A. C. Mandic, Recurrent neural networks for prediction: learning algorithms, architectures and stability., New York: John Wiley, 2001. |
| [27] | R. N. H. Kneser, "Improved backing-off for m-gram language modeling. In Acoustics, Speech, and Signal Processing," in *ICASSP-95*, 1995. |
| [28] | Q. V. L. a. T. Mikolov, "Distributed Representations of Sentences and Documents," *CoRR,* 2014. |
| [29] | V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady,* vol. 10, no. 8, pp. 707-710, 1966. |
| [30] | A. &. V. E. Marzal, "Computation of normalized edit distance and applications," *IEEE transactions on pattern analysis and machine intelligence,* vol. 15, pp. 926-932, 1993. |
| [31] | M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida.," *Journal of the American Statistical Association,* vol. 84, no. 406, pp. 414-420, 1989. |
| [32] | F. P. a. V. A. F. a. M. J. Miller, Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance, Alpha Press, 2009. |
| [33] | M. M. &. D. E. Deza, Encyclopedia of Distances, Springer, 2009. |
| [34] | N. X. Z. a. V. R. P. Evangelopoulos, "Latent semantic analysis: five methodological recommendations.," *European Journal of Information Systems,* vol. 21, no. 1, pp. 70-86, 2012. |
| [35] | T. e. a. Mikolov, "Distributed representations of words and phrases and their compositionality.," *Advances in neural information processing systems. ,* 2013. |
| [36] | P. Berkhin, Grouping multidimensional data, Springer, 2006. |
| [37] | S. Kaushik, An introduction to clustering and different methods of clustering, Analytics Vidhya, 2016. |
| [38] | J. P. J. &. K. M. Han, Data mining: concepts and techniques, Elsevier, 2011. |
| [39] | M. G. E. A. P. &. S. A. Omran, "An overview of clustering methods," *Intelligent Data Analysis,* vol. 11, no. 6, pp. 583-605, 2007. |
| [40] | S. &. K. K. Theodoridis, "Pattern recognition and neural networks," *Machine Learning and Its Applications,* pp. 169-195, 2001. |
| [41] | M. B. Y. &. V. M. Halkidi, "On clustering validation techniques.," *Journal of intelligent information systems,* vol. 17, no. 2, pp. 107-145, 2001. |
| [42] | P. H. &. S. R. R. Sneath, Numerical taxonomy, the principles and practice of numerical classification., W.H. Freemand and Company., 1973. |
| [43] | M. R. Anderberg, "Cluster analysis for applications," *Office of the Assistant for Study Support Kirtland AFB N MEX,* 1973. |
| [44] | B. Fritzke, "A growing neural gas network learns topologies," *Advances in neural information processing systems,* pp. 625-632, 1995. |
| [45] | J. Moor, "The Turing test: the elusive standard of artificial intelligence," *Springer Science & Business Media,* vol. 30, 2003. |
| [46] | "Axini," [Online]. Available: http://www.axini.com/model-based-testing/. |
| [47] | Qlikview. [Online]. Available: https://www.qlik.com/us/. |
| [48] | P. S. a. S. P. G. Efraimidis, "Weighted random sampling with a reservoir," *Information Processing Letters,* vol. 97, no. 5, pp. 181-185, 2006. |
| [49] | J. D. Musa, "“Operational Profiles in Software Reliability Engineering," *IEEE Software Magazine,* vol. 10, no. 2, pp. 14-32, 1993. |
| [50] | F. Mannhardt, M. de Leoni, H. Reijers, W. van der Aalst and P. Toussaint, "From Low-Level Events to Activities - A Pattern-based Approach," BPMcenter.org, Eindhoven, 2016. |