Future and Emerging Technologies
FET-Open

Phoenix

665347

Knowledge elicitation techniques analysis

Deliverable D3.1

v0.2

incas³ KU LEUVEN RWTH AACHEN UNIVERSITY TU/e

Document history – List of changes

Version	Date	Author name	Scope
v0.2	30/09/2016	Anil Yaman, Matt Coler, Giovanni Iacca	Final version
v0.1	26/09/2016	Anil Yaman, Matt Coler, Giovanni Iacca	Complete draft

Contents


Funded by the European Union

1 Introduction

WP3 deliverables contribute to the development of the Phoenix Knowledge Base and Human Interface Layer in Phoenix.

- **The Phoenix Knowledge Base (from D3.1, D3.2, and D3.3)** is where formal knowledge is stored. Developing the Phoenix Knowledge Base requires knowledge elicitation (D3.1) and knowledge representation (D3.2).
- **The human interface layer (HIL) (from D3.4 and D3.5)** is a medium for the knowledge exchange between the user and the Phoenix system. This will be described in more detail with the delivery of the two associated deliverables in M24.

This document serves as D3.1. It is a detailed manual, including a state-of-the-art survey, for identifying and eliciting expert knowledge. In Phoenix, expert knowledge is required for:

- identifying the requirements and goals of an exploration task obtained from a user;
- designing exploration procedures for answering to a user's question;
- interpreting the results of an exploration procedure, and
- presenting the results back to the user.

A knowledge elicitation process starts with identifying knowledge types and domains that are necessary for an application. In Phoenix, we require a procedure to achieve the objective of an exploration task given by a user's question. Therefore, we define procedural knowledge to represent these exploration procedures. Exploration procedures require knowledge in environments, motes, co-evolution and reincarnation; thus, we define this type of knowledge as domain knowledge.

We further define inference and empirical knowledge in Phoenix context. Inference knowledge allows drawing a conclusion over procedural and domain knowledge (see 2.2.3.3), and empirical knowledge is the formalization of the new knowledge acquired through exploration processes (see 2.2.3.4).

In this document, we first provide a literature review of the state-of-the-art methods in knowledge elicitation in section 2.1, and apply our methodology for knowledge elicitation to a demonstration case in section 2.2. We start with a user question, define the exploration procedure, and identify the domain knowledge involved into the exploration procedure. We have elicited knowledge involved in this example from the experts that are listed in section 4.2 based on the type of knowledge they contribute. Finally, the third offers a plan for the future work in knowledge elicitation. The definitions of the special terms we use in this document are provided in glossary 4.1.

2 Knowledge elicitation techniques analysis

Within Phoenix, the goal of knowledge elicitation is to acquire knowledge from experts to guide automatic reasoning. Phoenix is particularly dedicated to the use of expert knowledge to facilitate the exploration of unknown and inaccessible environments with motes that can provide information about an environment and answer a user's question.

As Phoenix involves knowledge from multiple scientific fields, the knowledge elicitation techniques we employ require:

- methods to elicit various types of knowledge from different domains;
- the cooperation of experts from relevant domains.

The next section we provide a state-of-the-art survey in knowledge elicitation.

2.1 State-of-the-art survey in knowledge elicitation

A step-by-step guide to knowledge elicitation processes for knowledge engineers is provided by Milton (Milton, 2007). According to this view, knowledge elicitation has three steps: 1) identify the domain/s of investigation, 2) select experts with relevant knowledge, and 3) perform knowledge elicitation, as below:

Step 1. Identify domain of investigation: In Phoenix, the process of identifying domains starts with determining the requirement/s relevant to a user's questions. They can be scientific domains (in the case of Phoenix: earth science, hydrology, fluid dynamics, etc.), research domains (in the case of Phoenix: knowledge representation, Artificial Evolution, sensor technology etc.), domains of practice (in the case of Phoenix: mapping, engineering, simulations, etc.), or some combination thereof.

Step 2. Select experts: It is ideal to involve multiple experts if there are conflicting views. Expertise is not something easy to identify, especially for cases in which the knowledge engineer lacks the expertise in question. Generally speaking, experts are individuals who are highly regarded in a community of peers because of their ability to produce uncommonly accurate and reliable judgments made with skill and economy of effort. Experts are capable of dealing with tough/unusual cases thanks to special skills and knowledge arising from extensive experience. According to a rule of thumb, it takes about 10,000 hours of practice to develop expertise (Shadbolt & Smart, 2015).

Involving experts is important because of well-known differences between classifications made by novices and those made by experts. It is also crucial to recognize the type of expert that a knowledge engineer works with, because different experts have different kinds of expertise which are elicited and represented in different ways. Shadbolt and Smart categorize three types of experts: the academic, practitioner and samurai (Shadbolt & Smart, 2015):

- The *academic* experts possess logically consistent and coherent knowledge in their domains. They value theoretical generalizations, and believe that problems are instances of these generalizations; however, the academics may be remote from every day-to-day problem solving.
- The *practitioners* are mainly involved in day-to-day problem solving. Their knowledge is often implicit and involve heuristics in problem solving that are usually acquired from earlier solved problem cases.
- The *samurai* type is performance expert. These experts are usually trained only by practice and knows how to optimally perform a process. They may not have a theoretical understanding of the domain. Therefore, they may not be capable of providing the logic behind their expertise (Shadbolt & Smart, 2015).

These types of experts can be found in medical domain where academic types are the professors, practitioners are the doctors actively providing medical treatment, and the samurai type is the medical staff that perform repetitive clinical tasks (Shadbolt & Smart, 2015).

Step 3. Knowledge elicitation: in this step, knowledge elicitation methods are used to elicit knowledge from identified experts[1]. There are a number of methods described in the literature. A comprehensive survey of the knowledge elicitation methods in the literature can be found in (Shadbolt & Smart, 2015), (Milton, 2003). These methods are especially designed for eliciting the relevant types of knowledge[2].

To present purposes, knowledge can be divided into domain knowledge and procedural knowledge.

- **Domain knowledge:** knowledge of a certain topic, typically called "facts", for which there is typically a high degree of consensus between experts.
- **Procedural knowledge:** knowledge on how to perform a task or tasks in an optimal or efficient way. Also known as know-how.

Domain and procedural Knowledge can be explicit or implicit. **Explicit knowledge** is knowledge that can easily be communicated through language. Examples include knowledge in scientific fields and facts about the world. **Implicit knowledge**, on the other hand, is difficult to communicate in language. Examples include knowledge on how to ride a bike, become an expert chess player, or a chef.

A Similar categorization of knowledge is made by the common knowledge acquisition and documentation structuring (CommonKADS) model (Schreiber et al., 2000). CommonKADS is a knowledge engineering model that aims to aid knowledge engineers in distinguishing between different knowledge types involved in applications. CommonKADS defines inference knowledge in addition to the domain and procedural knowledge. Inference knowledge allows drawing conclusions over procedural and domain knowledge. In this model, the domain knowledge forms the basis for the inference knowledge, and procedural knowledge uses these inferences to achieve the goals of a task.

[1]Multiple experts from the same domain may participate in the knowledge elicitation process. This elicitation process is often referred to as collaborative knowledge elicitation. Collaborative knowledge elicitation is preferable to knowledge elicitation from a single expert.

[2]In the knowledge elicitation process, a bottom-up, top-down or both approaches can be adopted. The knowledge elicitation process with a bottom-up approach starts by defining the concepts from the most to the least specific; whereas, a top-down approach starts defining concepts from the most to the least general. An approach that combines both aims to define the most general and specific concepts in a parallel fashion, and iteratively define the concepts in the middle to reduce the gap between the top and bottom levels.

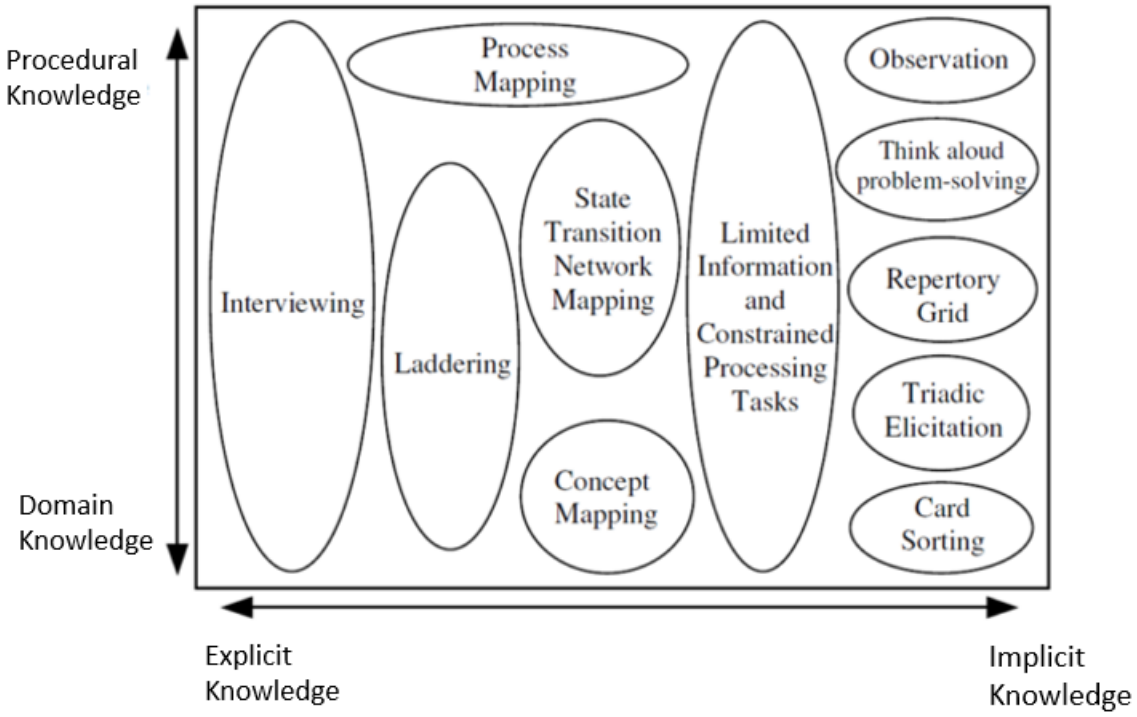


Figure 1: Knowledge elicitation methods and their suitability of type of knowledge in two dimensions (taken from (Milton, 2003)).

Figure 1 illustrates several knowledge elicitation methods corresponding to the aforementioned types of knowledge. A common method is interviewing (Geiwitz, J and Kornell, J and McCloskey, 1990; Hudlicka, 1997), where knowledge engineers or experts interview fellow experts. The interviews can be structured, semi-structured or unstructured. Interviewing techniques are good for eliciting explicit domain and procedural knowledge (Ericsson & Simon, 1984); however, they may not be ideal for eliciting implicit knowledge. For implicit knowledge, methods such as observation (Belkin & Brooks, 1987) and thinking aloud problem-solving (Ericsson & Simon, 1984) are common methods, especially for procedural knowledge. Card sorting is common for capturing implicit knowledge. We give a detail description of interviews, observation, thinking aloud problem-solving, card sorting, and triadic elicitation in Appendix 2.

2.1.1 Uncertainty in knowledge elicitation

Elicitation techniques should be able to handle uncertainties that may be present in expert knowledge (O'Hagan, 2005). In this context, uncertain knowledge refers to the unknowns and/or estimates in the knowledge elicited from the experts. These uncertainties may arise from stochastic or incompleteness of expert knowledge, or as an outcome of collaborative knowledge elicitation (Albert et al., 2012).

The elicitation of uncertain knowledge often requires experts to quantify the uncertainty in their knowledge. However, it is usually difficult for experts to do this. It is often more natural to use modal quantifies, such as: *could*, *may*, *might*; or vague terms, such as: *probably*, *likely* (in English). Therefore,

knowledge engineers should be aware of this and ask for elaboration to determine "where" the uncertainty is, and probe the limits of the expert's certainty.

Using uncertain quantifier in language can only capture explicit knowledge provided by the answers obtained from the experts in a natural language form. Uncertain implicit knowledge on the other hand can be captured by observing the variations in a process or task.

2.2 Methodology

In this section, we describe our methodology for eliciting knowledge in Phoenix. In accordance with the steps mentioned previously, we start by identifying the domains of investigation in 2.2.1. Next, we identify relevant experts in 2.2.2. Finally, we describe the knowledge elicitation methods used in knowledge elicitation in 2.2.3 based on the relevant knowledge domains in 2.2.1.

2.2.1 Identify domain of investigation

We distinguish between four types of knowledge relevant to the Phoenix system. In addition to the aforementioned domain and procedural knowledge, we introduce two other kinds of knowledge relevant to our purposes: empirical and inference knowledge. The knowledge types are described below in terms relevant to Phoenix:

- **Domain knowledge** includes the explicit facts related to the environment types, motes, mote components and trade-offs, co-evolutionary and reincarnation cycles, and their relations.
- **Procedural knowledge** defines the expertise involved in performing the exploration tasks.
- **Empirical knowledge** refers to the heuristic nature of expert knowledge that is used in some domain or process. This type of knowledge is specific to the Phoenix approach. This type of knowledge can be gained or refined through observation, senses and/or experiences (see 2.2.3.4). Empirical knowledge may or may not consist of established facts, rather it constitutes rules of thumbs that can work in specific cases. These rules of thumbs can either initially set by an expert (likely *practitioner type*) or acquired during the process. This type of knowledge is valuable for problem-solving processes.
- **Inference knowledge** type of knowledge that arises from domain, procedural and empirical knowledge through inference processes.

The empirical knowledge is defined to reflect the empirical aspect of the Phoenix approach. It aims to answer the user's question about an unknown environment by using hypothesizing and verifying steps that involves conducting experiments on the co-evolutionary and reincarnation cycles. The empirical knowledge develops the knowledge that is based on the observations and experiences obtained during this problem solving process. This knowledge can easily be applied to the problems as heuristics to help with the problem solution process. Therefore, empirical knowledge is separately defined as a specific type of knowledge layer in Phoenix.

The knowledge types presented here are identified based on the requirements of Phoenix. First, the procedural knowledge derived from the user's question. The procedural knowledge refers to the set of experiments (co-evolution and reincarnation) to provide the answer to the user's question. From procedural knowledge, we can identify the required domain knowledge.

In section 2.2.3, we start elaborate a use case demonstration of knowledge elicitation process. The example demonstration we work on starts with a user's question: "Where is the location of a t-junction?" In order to elicit procedural knowledge for this problem, it is required to find experts with procedural knowledge in this process. Section 2.2.2 provides information about selected experts.

Elicitation results of the procedural knowledge for this problem is given in section 2.2.3.2. Once the procedural knowledge is complete, we identified the domain knowledge involved in the procedural knowledge, and provided our results in section 2.2.3.1.

2.2.2 Expert selection

Section 4.2 provides an overview of the experts that contribute knowledge to Phoenix. The experts were classified based on the type of knowledge they possess. Each type of expert(s) is identified by an **ID** that assigned to each row. The process of required domain, inference and procedural knowledge elicitation are likewise conducted based on the type of knowledge that the experts possess.

The expert selection process is performed for the demonstration example provided Table 1. We first identified experts with procedural knowledge in the exploration procedures. Even though expert **IDs 1, 2, and 4** do not have specific expertise with exploration in pipeline, they possess knowledge on the exploration tasks in other types of environments such as: flume or oil reservoir (E Duisterwinkel, Krijnders, Talnishnikh, van Hengel, & Wörtche, 2016; Talnishnikh, van Pol, & Woertche, 2015). Therefore, we conducted interviews with the expert **IDs 1, 2, 3, and 4** for eliciting knowledge on the experiment processes in pipelines using motes.

Using the procedural knowledge, we identified the following domains: localization, accelerometer, co-evolution, reincarnation, t-junction pipeline and environment classification involved into the demonstration example. Below, we provide the source and extend of the expert knowledge used in the demonstration:

- For eliciting the domain knowledge in localization, we had interviews with expert **ID 2** and discussions with expert **ID 5**. They have provided the properties of localization and required optimization objective in co-evolutionary cycle (see section 2.2.3.1).
- The domain knowledge in accelerometer sensor used in the demonstration was provided by the expert **ID 6**.
- The domain knowledge in co-evolutionary cycle was provided by the experts given in row **ID 7**.
- The domain knowledge in reincarnation cycle was provided by the experts given in row **ID 1 and 2**.
- The domain knowledge in the t-junction pipeline environment and environment classification (see 2.2.3.1) was provided by the experts **ID 1, 2, and 3**.

2.2.3 Knowledge elicitation

We used interviewing and thinking aloud problem-solving methods during the knowledge elicitation processes we have performed so far. We mainly applied collaborative knowledge elicitation techniques. The experts have come to an agreement of the result of the collaborative knowledge elicitation process.

This example we work is a special case of an exploration task (e.g. considering only 2-dimensions) defined collaboratively by the experts participating in the project to build an initial prototype which is scalable to the complete Phoenix functionality.

Table 1: The problem definition of an exploration task.

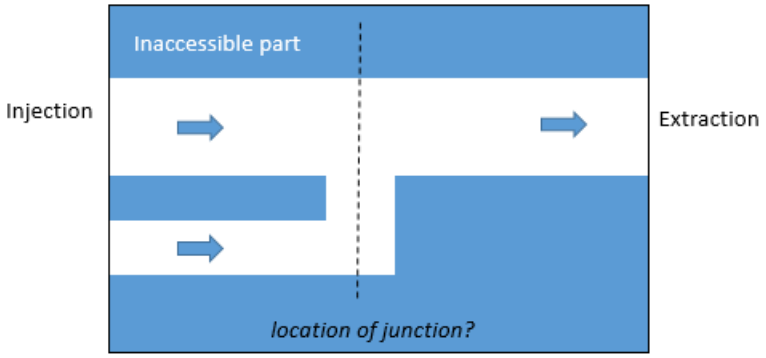
Environment:	 **Figure 2:** Simple visualization of a t-junction environment.
User's Question:	Where is the location of the t-junction in a pipeline?

Table 1 presents a visual representation of the environment, and the user's question. In this example, the user is interested in knowing the location of the t-junction in a pipeline. As discussed in section 2.2.1, the Phoenix approach for solving this exploration task involves domain, procedural, inference and empirical knowledge. In the next sections, we present the knowledge involved into each of these knowledge types.

For clarity, we strictly limit our scope of knowledge elicitation process to the domain, procedural, inference and empirical knowledge within the scale and requirements given in the problem definition presented in Table 1.

Note, moreover, that this deliverable provides a preliminary report on the current state of the knowledge elicitation process. The knowledge elicitation results presented here are subject to iterative refinement. An outline of future work is provided in 2.3.

2.2.3.1 Domain knowledge

We present our knowledge elicitation results in environments, motes, co-evolutionary and reincarnation cycles separately in detail.

Domain knowledge in environments and their properties:

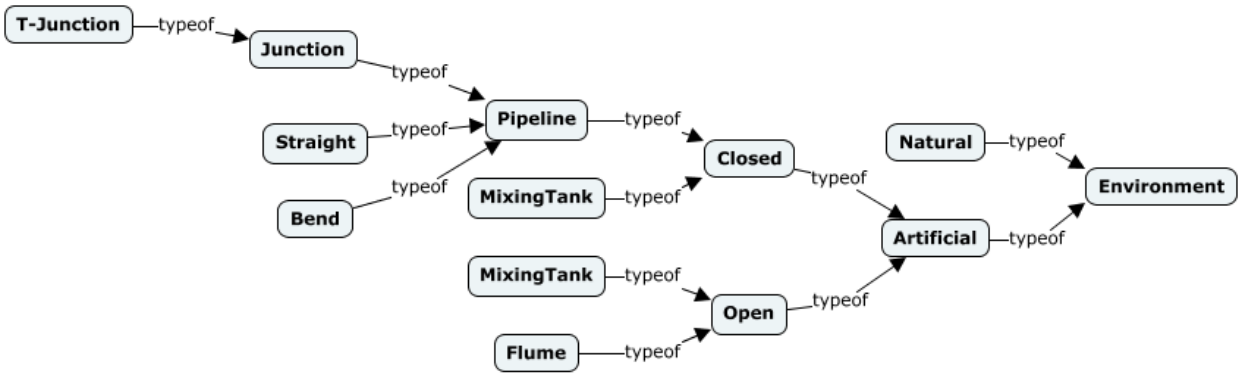


Figure 3: Types of environments, with focus on pipeline environments.

Phoenix is suitable to apply to a wide range of environments including pipelines, flumes, mixing tanks, rivers, underground rivers, underground caves, potash mines, oil reservoirs, etc. In this deliverable, we focus on a t-junction pipeline. Figure 3 illustrates our classification of the environments. Each node in the figure is labelled with the name of an environment, and each link defines the semantic relations between these environments. In this example, the only relation provided is "typeof". This relation indicates the hierarchical relationship between the parent and child concepts where child concepts possess more specific properties than their parents. Moreover, the parent and child concepts share common properties, and the sibling concepts may or may not share common properties. The definitions of the concepts shown in the figure are given in Table 2.

Table 2: The definitions of the concepts illustrated in Figure 3.

Concept	Definition
Environment:	The surroundings and conditions in which motes operate.
Artificial:	Environment that is constructed by humans.
Natural:	Environment that emerged due to natural causes or processes.
Closed:	Environment that does not have a free flow surface.
Open:	Environment that has a free flow surface.
Mixing Tank:	Machine controlled artificial environment that is used to blend various liquids together.
Flume:	An artificial channel conveying liquid.
Pipeline:	A line of pipe conveying liquids.
Straight:	Type of pipeline environment that extending uniformly in one direction.
Bend:	Type of pipeline environment that shaped as a curve.
Junction:	A pipeline environment where two or more pipes joined.
T-Junction:	A pipeline environment where a pipe joins another without crossing it.

Domain knowledge in motes, mote components, and their relations to the environmental properties:

Figure 4 shows the domain knowledge elicited for the example exploration task given in Table 1. In the figure, the concepts and relations are shown by nodes and edges similar to the one provided for environment types. The node labels provided in bold and with a grey background represent concepts, and the rest of the labels represent the properties of the concepts. In this example, the concepts are categorized under three groups: environment, mote and functional concepts. Environment concepts describe the environment, mote concepts describe mote and mote components, and functional concepts define two concepts that play role in mapping velocity profile of the environment.

The relations between concepts and properties are shown by directed edges (link between vertices). The arrows represent the direction of relation. Each edge is labelled by the semantic of the relation. These semantics are used in their intentional meanings. For example, directional relation **hasInjectionDiameter** connects the environment **T-Junction** to its property **Di**. This example demonstrates that t-junction environment has a diameter value. Similarly, a mote must have a physical sphere shape which also has a diameter value. Moreover, the diameter value of a mote must be smaller than the diameter of the injection and extraction t-junction pipe; because the contrary would not allow injection and extraction of motes into and out of the t-junction. Although, the demonstrated semantic relations can exist between the concepts, it does not mean that an instantiation of a concept must have all the properties initialized; unless otherwise is indicated (e.g. relation **mustHaveBattery**).

We can define a specific mote or environment by assigning specific values to their properties. This process generates an instance of a class. We refer specific instantiations of motes and environments as mote and environment instances. The instantiation process can be performed for any type of knowledge represented formally in the Knowledge Base.

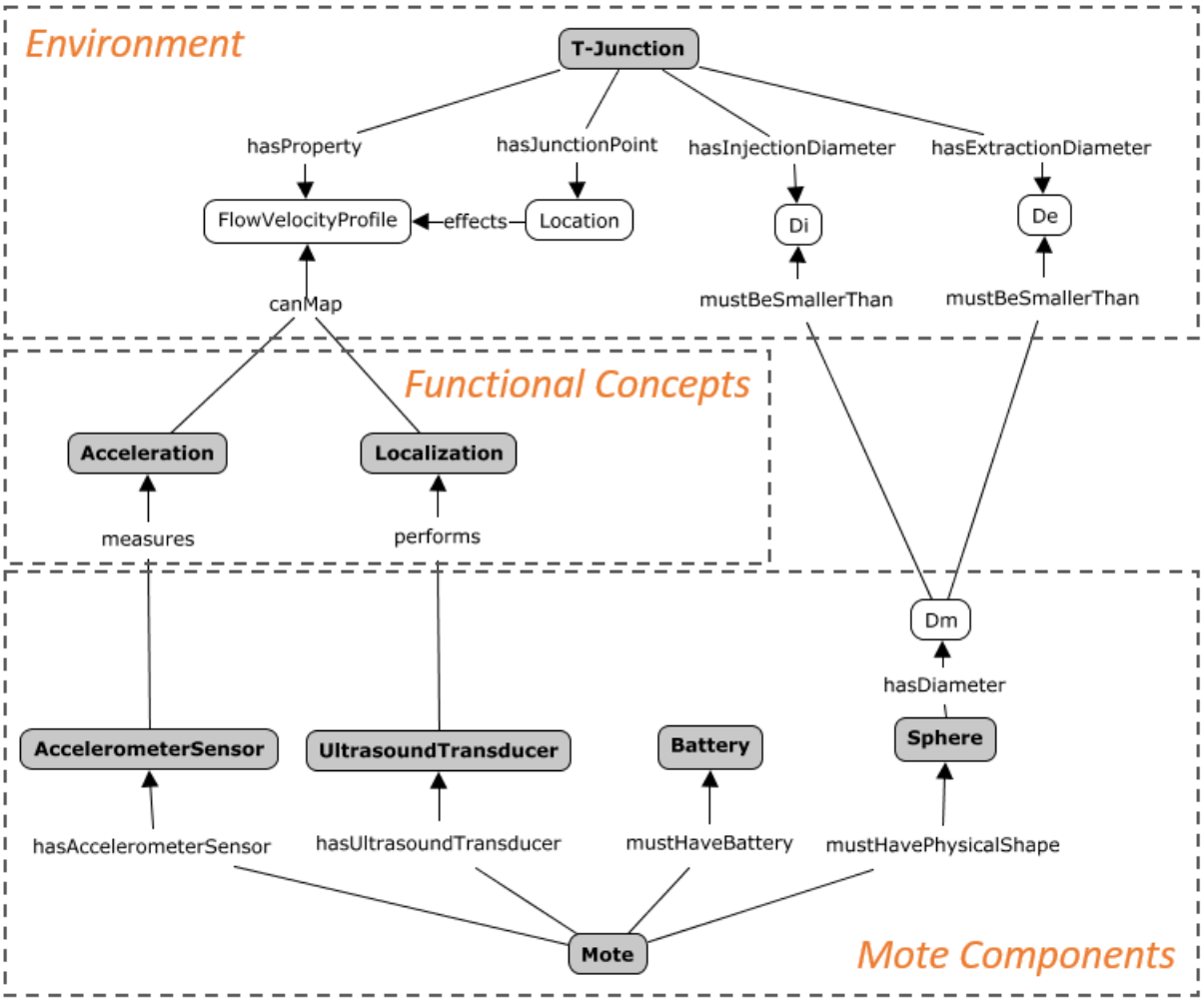


Figure 4: The domain knowledge in mote and t-junction environment, and their relations.

Figure 4 shows that using **Accelerometer** and **Localization** the flow velocity profile of the environment can be mapped. **Accelerometer, Localization,** and **UltrasoundTransducer** have properties that are not shown on the figure for simplicity. These properties are shown in **Table 3**. According to the table, **Localization** associated with two properties, **NumberOfMotes** and **Range**. The evolvability of these properties is further indicated in the table.

Table 3: Properties of Concepts in Figure 4.

Concept	Property	Evolvable
Localization	NumberOfMotes	Yes
Localization	Range	Yes
UltrasoundTransducer	Frequency	Yes
AccelerometerSensor	SamplingFrequency	Yes

We use the exploration task defined in **Table 1**; however, we consider only the location property of the environment as unknown by the user which is posed as the initial question. The properties **Di** and **De** of the environment can simply be requested from the user or provided by the Knowledge Base. In either case, we assume that these properties are known.

Domain knowledge in co-evolutionary cycle:

The concepts and properties in co-evolution were elicited from experts given in section 4.2. These concepts are summarized in Table 4. Under the column header "Co-evolutionary Property Names" the properties of the co-evolutionary cycle are listed. These properties define function, input, and output of the co-evolution cycle.

The function property of co-evolution defines its utility. It is necessary to know what co-evolution can do in order to apply when the necessary functionality is required. We limit the utility of the co-evolution to our demonstration case and take into account only evolution of mote component and method parameters (ultrasound transducer, accelerometer, and localization); however, co-evolution in the Phoenix approach considers many more additional mote components, methods, and co-evolving models of environments. These additional functions will be added when the initial demonstration is extended to other problem cases. Never the less, we will refer to the evolutionary process of motes in our special case as "*co-evolution"*. In short, the utility of co-evolution is to optimize the properties of ultrasound transducer, localization and accelerometer to satisfy objectives.

The rest of the properties of co-evolution are the **Environment**, **Mote**, **Objective**, **Constraints**, **EvaluationMethod**, and **Output**. The co-evolution requires initializing the first generation of motes "*Gen-0*" (Proposal, 2014) by instantiating the properties of mote components and methods. The initialization process can be performed by either user or Knowledge Base. How the initialization should be done is part of domain knowledge and empirical knowledge (see 2.2.3.4).

The objective of the co-evolutionary cycle may depend on the user's question. Here, we only consider optimizing the properties of **Localization, Accelerometer,** and **UltrasoundTransducer**. We have a very limited power, and it is drained by each action (e.g. sampling, storing, etc..) (Xin et al., 2015). Therefore, we use energy budget as a constraint. We further provide the total consumed time for simulation to the co-evolution as a constraint since we expect the total computation of simulations to finalize within a reasonable time.

Table 4: The domain knowledge in co-evolution cycle.

Co-evolution Property Names	**Property Definition**	**Property values specific to the example case**
Function:	- Optimizes the properties	- Localization - Ultrasound transducer - Accelerometer
Environment:	- Requires the environment type and its properties for simulation	- T-Junction with only the location of the junction is not known by the user

Mote:	- Required for initialization of mote instances	- Mote properties for initialization
Objective:	- Required for the objectives of co-evolution in selection phase	- Ability to construct visual environment from ultrasound readings as accurate as possible
Constraints:	- Required input for constraint types	- Battery power - Total consumed time in simulations
EvaluationMethod:	- Required for evaluation	- ComparativeMethod (Comparison of the localization results of the co-evolutionary and reincarnation cycle)
Output:	- The results of co-evolutionary cycle	- Localization results - Fittest mote instances - Accuracy of the localization - Statistics of the mote instances (e.g. power consumption)

The evaluation method refers to the approach used for evaluating the mote instances. There may be several approaches that can be used in the evaluation; thus, the definitions of evaluation approaches belong to the domain knowledge in co-evolutionary cycle. However, in this example we will demonstrate the approach that compares the localization results of the motes on simulation and real environments for determining their fitness values. In real case, we cannot use this approach; because the properties of the real world environment are unknown. Instead, we use a co-evolutionary approach which will be described in detail in WP4.

Finally, the output property of the co-evolutionary cycle defines the expected outcomes of this process. These outcomes are the localization results of the fittest agent instance, the properties of the fittest mote instance, the accuracy of the localization, and the statistics of the mote instances that are collected during the co-evolutionary cycle.

Domain knowledge in reincarnation cycle:

The domain knowledge in the reincarnation cycle is shown in Table 5. The reincarnation cycle has four properties named as **Function**, **Environment**, **Mote** and **Output**. **Function** defines its utility. It is to collect the data of a performed experiment in the real world. A real world experiment involves real physical motes. However, we define the reincarnation cycle based on the current resources we have for the example exploration task. Since we currently work on building a real world environment and physical mote hardware to conduct real world experiments, we use a simulated environment in reincarnation cycle and assume that it is the real world environment.

Table 5: The domain knowledge in reincarnation cycle.

Reincarnation Property Names	Property Definition	Property values specific to the example case
Function:	**-** To demonstrate the functionality of the motes in real world environment by produce the data gathered during the process	**-** Localization - Ultrasound transducer - Accelerometer
Environment:	- A prefined model of an environment	- t-junction environemnt that all of its properties are known (they are arbitrarily defined a priori in order to simulate a real world environment)
Mote:	- Required input for properties of mote components and methods	- Mote properties for initialization
Output:	- Data gathered from a real environment	- Localization results - Accelerometer

The properties of the real world environment involve an environment, mote and output. The environment is a simulation where all of its properties are known. Mote defines mote properties that can be initiated and send to the environment; and the output is the localization results and accelerometer sensor readings are extracted from the motes.

2.2.3.2 Procedural knowledge

In Phoenix, the procedural knowledge defines the expertise involved in the exploration processes to contribute to answering a user's question. During the knowledge elicitation process for procedural knowledge, we collaborated with the experts in exploration processes (see 4.2). We used thinking aloud problem-solving methods to elicit consensual procedural knowledge collaboratively.

As a result, we extrapolated a process to address exploration-related question provided in Table 1 for the environment visualized in Figure 2. Table 6 provides our results. Procedural knowledge can be formalized hierarchically. According to this formalization, procedural knowledge may be comprised of other (procedural, domain, or inference) knowledge. Each row in the table defines an action step in a sequential order given by the first column. The final column lists the type of knowledge most pertinent to each step.

Table 6: Procedure for solving the problem given in Table 1.

#	Action Step	Explanation of the output of the action step
1)	Request the user's qestion.	
2)	Obtain the user's question.	User's question is "where is the location of the t-junction in a pipeline?".
3)	Identify the type of the environment that user would like to investigate.	Type of environment is identified from the user question. Domain knowledge given **Figure 3** is used to recognize "**T-Junction**"

4)	Identify the property of the environment the user is interested in.	The user interested in finding the **Location** property of the **T-Junction**. This is found by interpreting the user's question and the domain knowledge given in **Figure 4**.
5)	Identify how the interested property of the environment can be found.	Looking at the **Figure 4**, it is seen that the **Location** of a **T-Junction effects** the **FlowVelocityProfile.** Therefore, if **FlowVelocityProfile** is obtained, we can show where the **Location**.
6)	Identify the mote components that are required for finding interested environment property.	Looking at the **Figure 4, Acceleration** and **Localization** are needed to find **FlowVelocityProfile.** - For **Acceleration,** Mote should have **AccelerometerSensor** - For **Localization**, Mote should have **UltrasoundTransducer** - Mote also has **Battery** and **Sphere PhysicalShape**
7)	Identify the properties to initiate motes.	From **Table 3**, properties of **Localization**, **UltrasoundTransducer**, and **AccelerometerSonsor** are found Battery Power assumed Fixed From **Figure 4**, Mote has a Diameter property **Dm**
8)	Identify how to initialize mote properties.	From **Table 3, Localization** properties **NumberOfMotes** and **Range** are optimized by co-evolution, and **AccelerometerSensor** property **SamplingFrequency** is optimized by co-evolution
9)	Identify what is needed to initialize co-evolution	From **Table 4,** co-evolution requires initialization of its properties: **Environment, Mote, Objective, Constraint, EvaluationMethod**
10)	Initiate properties of co-evolution	
11)	Identify the properties of Environment to initiate	From **Figure 4,** we can find that we need injection and extraction diameters **Di** and **De** of the Environment. **Location** is not known (and interested by the user) and **FlowVelocityProfile** will be mapped (to answer the question).
12)	Identify method to initialize properties	The initial values for the properties can be retrieved from the Knowledge Base, asked to the user, or initialized randomly
13)	To be Developed further...	

We label the type of knowledge experts use in each procedural step. For example, in step 3), experts use inference knowledge to identify the most likely kinds of environment implicit in the user's question. This inference is made based on their domain knowledge in the environment types as it was shown in section 2.2.3.1. It is likely that the experts recognize the concepts in the user's question and establish the asso-

ciation between their knowledge of the environment. The inference knowledge uses domain knowledge to draw conclusions (see 2.2.3.3). As mentioned, a similar process is applied when the experts use the domain knowledge in a step. For example, in step 4), they directly relate from the domain knowledge in the environment that the t-junction environment has a junction point.

2.2.3.3 Inference knowledge

Based on our observations, the experts often apply a reasoning process during an exploration task. In a most general sense, experts use this type of knowledge to generate hypotheses based on the available knowledge, then test these hypotheses by conducting a set of experiments. Based on the outcome of the experiments, they update the available knowledge, and repeat the process by starting from the hypotheses generation process until a satisfactory answer is found to the user's question. The inference knowledge plays a crucial role in this process. It uses the domain knowledge and the user's question to generate hypotheses, and help generating a set of experiments to test these hypotheses. Table 6 steps 5) and 6) show examples of the inference knowledge where how the user's objective can be achieved, and what mote components required for this process are identified respectively.

The role of the inference knowledge is not only limited to the main roles that are mentioned above. Expert also use inferences in a much smaller scale tasks that require drawing conclusions based on the presence of available knowledge. For example, in step 3), experts recognize the environment type that the user is interested in. This recognition step requires making inferences based on the concepts in the user's question and the domain knowledge about the environment to correctly identify what kind of environment the user is interested in investigating.

Experts may use various methods to perform inference process. These methods can be deductive, inductive or abductive. **Deductive inference** can infer the effect given the cause and the rule. It guarantees the that the result is true if the cause and rule are true and sound. For example, if junction in a pipeline effects flow velocity profile (rule), and there is a junction in the pipeline (cause), then the effect of the junction must be seen in the flow velocity profile (effect).

Inductive inference can infer the rule given the cause and effect. For example, if there is a junction in the environment (cause), and there is an effect in the flow velocity profile (effect), then we can infer that junction effects the flow velocity profile (rule). It is not ideal to perform this type of inference based on only one instance of cause and effect. Although, it is possible to interpret the reliability of inductive inference in the context of probability theory, it cannot guarantee the truth of the inference.

Abductive inference can infer the cause given the rule and effect. In this case, if a junction in a pipeline effects the flow velocity profile (rule) and there is an observable effect in the flow velocity profile (effect), then we can infer that there should be a junction (cause). This type of inference is the most intuitive in exploration tasks where the experts usually use the rules in their domain knowledge and observe the effects of the experiments to infer a cause. There may usually be a number of causes that can explain the effects. Some of them may be more probable than others (probability can be assigned based on how many effects a cause can explain). Then the most probable cause should be accepted. It may be the case that there may be multiple number of causes can best explain the effects. However, it is preferable to explain as many as effects using less number of causes as possible for simplicity (a.k.a. Ockham's Razor, see ("Simplicity", 2016)).

2.2.3.4 Empirical knowledge

We define empirical knowledge to refer the type of expert knowledge that is acquired by experience, observation and/or senses. Empirical knowledge is mainly based on heuristics. These heuristics are approaches to problem solving that can be updated based on new experience. This type of knowledge is not apart from the procedure, domain or inference knowledge. It is rather a dynamic component in these knowledge types it develops over time.

An example of developing a heuristic can be applied step 8) in procedural knowledge given in Table 6. If the expert is encountered with this exploration task for the first time, he/she can store the initialized property values by requesting from the user. Next time when a similar experiment is performed and if the user does not know what to initialize for these properties, the stored values can be better choice for initialization step rather than random initialization since it is likely to improve the performance. The heuristic we develop in this case can be stated as "use the property values initiated before in a similar experiment if the user does not know what to initiate for the current exploration task".

2.2.4 Generalizability and extendibility of knowledge elicitation

The structure of knowledge we obtain as a result of our knowledge elicitation process is presented in section 2.2.3. As can be seen in the structure of knowledge (e.g. Figure 3 and Figure 4), the concepts and their relations involved in the current knowledge can easily be monitored and modified. This makes it easy to extend the current knowledge with additional knowledge.

The example exploration task describes one special instance of procedural knowledge on a specific environment. This will not be the case when we extend the procedural knowledge with different kinds of procedures applied on different types of environments. After all, it would not be possible to manually extend and/or manipulate the knowledge structure of all possible exploration procedures for all possible environments and objectives. Rather, a generic approach will be used. This approach will make use of generalizations, templates and hierarchical definitions of procedural knowledge. Moreover, it may be possible to transfer and use the knowledge (that is, use similar exploration processes for similar environments) between similar exploration tasks.

2.3 Future work

In this section, we discuss our plan for extending the knowledge elicitation process to the full scale of the Phoenix approach building upon the results we achieved in this deliverable. The framework of knowledge elicitation is modular; thus, extensions and maintenance are quite easy to make. Some possible strategies for extension of knowledge elicitation are discussed in section 2.2.4. We plan on continuing the knowledge elicitation process in different types of knowledge in parallel. We firstly aim to make use of available knowledge in motes and mote components that will be elicited mainly from the experts as well as other resources that are delivered by experts during the first year of the project. Such tutorials and reports are readily available for knowledge extraction and use (Erik Duisterwinkel & Xin, 2015; Harpe, 2015; Iacca, n.d.; Phoenix, 2016; Xin et al., 2015). The current exploration task involves only the objective for optimizing the localization process and battery usage as efficient as possible. Additional definitions associated with the objectives, tradeoffs and constraints (e.g. production cost of a mote) are planned to be added in next phases of knowledge elicitation.

We are elaborating an inventory of the required knowledge (domain, procedural, inference and empirical) for the demonstration. We will work with the consortia to elaborate the (pseudo-)user's questions for the demonstration, simulating the role of the HIL (which isn't developed until M12). Based on the parameters of that question, we will identify the knowledge relevant to addressing it. We expect that some of the experts who have that knowledge will come from outside the consortia. Therefore, we will soon apply the knowledge elicitation techniques with external experts. This knowledge will be formalized as per D3.2 and integrated into the Phoenix Knowledge Base. Future experiments will be dedicated to determining how little knowledge is enough to answer users' questions.

The current evaluation approach in co-evolutionary cycle is described in section 2.2.3.2. The method uses a comparative approach which may not be applicable to the real world environments (assuming the properties of the real world environment are not known). Therefore, the next logical thing is to initiate the elicitation process with the experts on the evaluation methods that are suitable to the Phoenix approach. This process is eventually expected to lead to the co-evolution of environment models. Therefore, the processes of eliciting and incorporating the knowledge in evolvable properties of environments are necessary. Moreover, real world environment uses a pseudo real environment (currently we use a simulated environment instead of a real one for comparison). Therefore, the definition of the environment in reincarnation cycle will be replaced with a real environment.

The knowledge elicitation method we use now does not involve uncertain knowledge elicitation. Eventually, we will include methods to elicit uncertain knowledge.

3 Reference

Albert, I., Donnet, S., Guihenneuc-Jouyaux, C., Low-Choy, S., Mengersen, K., & Rousseau, J. (2012). Combining expert opinions in prior elicitation. *Bayesian Analysis*, *7*(3), 503–532. http://doi.org/10.1214/12-BA717

Belkin, N. J., & Brooks, H. M. (1987). Knowledge elicitation using discourse analysis, 127–144.

Cordingley, E. S. (1989). Knowledge elicitation techniques for knowledge-based systems. *Knowledge Elicitation: Principle, Techniques and Applications*, 87--175.

Duisterwinkel, E., Krijnders, D., Talnishnikh, E., van Hengel, P., & Wörtche, H. (2016). Feasibility Study XWM in Flume IWW RWTH Aachen , 4 th July 2016, (July).

Duisterwinkel, E., & Xin, H. (2015). Ultrasound Signal Processing. Retrieved from http://www.phoenix-project.eu/

Ericsson, K. A., & Simon, H. . (1984). *Verbal reports as Data*. Cambridge, MA: The MIT Press.

Geiwitz, J and Kornell, J and McCloskey, B. (1990). An expert system for the selection of knowledge acquisition techniques. *Santa Barbara, CA: Anacapa Sciences*.

Grover, M. (1983). A pragmatic knowledge acquisition methodology. In *8th International Joint Conference on Artificial Intelligence*. Karlsruhe, Germany.

Harpe, P. (2015). Timing Tutorial. Retrieved from http://www.phoenix-project.eu/

Hoffman, R. R. (1987). The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology. *AI Magazine*, *8*(2), 53–67. http://doi.org/10.1609/aimag.v8i2.583

Hudlicka, E. (1997). Summary of knowledge elicitation techniques for requirements analysis. *Course Material for Human Computer Interaction, Worcester Polytechnic Institute*.

Iacca, G. (n.d.). Phoenix - D2.1.

Milton, N. R. (2003). *Personal Knowledge Techniques*. University of Nottingham, Nottingham, England, UK.

Milton, N. R. (2007). *Knowledge Acquisition in Practice: A Step-by-step Guide (Decision Engineering)*. Retrieved from http://www.amazon.com/Knowledge-Acquisition-Practice-Step-step/dp/1846288606

O'Hagan, A. (2005). Research in Elicitation. In S. K. Upadhyay, U. Singh, & D. K. Dey (Eds.), *In Bayesian Statistics and its Applications* (pp. 375–382). Anamaya: New Delhi.

Phoenix. (2016). D2.2 - Specifications of the work packages. Retrieved September 1, 2016, from http://www.phoenix-project.eu/

Proposal, P. (2014). FET Phoenix Proposal- Exploring the Unknown through Reincarnation and Co-evolution. Retrieved from http://www.phoenix-project.eu/

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. De, Shadbolt, N. R., & Wielinga, B. (2000). Knowledge Engineering and Management: The CommonKADS Methodology. *Knowledge Creation Diffusion Utilization*, *99*, 455. http://doi.org/10.1016/S0933-3657(01)00090-2

Shadbolt, N. R., & Smart, P. R. (2015). Knowledge Elicitation : Methods , Tools and Techniques. In *Evaluation of Human Work* (p. 43). CRC Press.

Simplicity. (2016). Retrieved from http://plato.stanford.edu/entries/simplicity/

Talnishnikh, E., van Pol, J., & Woertche, H. J. (2015). Micro Motes: A Highly Penetrating Probe for Inaccessible Environments. In H. Leung & S. Chandra Mukhopadhyay (Eds.), *Intelligent Environmental Sensing* (Vol. 13, pp. 33–49). inbook, Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-12892-4_2

Xin, H., Harpe, P., Gavannarapu, S., Verhelst, M., Sadeghpour, S., & Puers, B. (2015). Hardware and energy consumption tutorial. Retrieved from http://www.phoenix-project.eu/

4 Appendix

4.1 Glossary

Collaborative knowledge elicitation: the process in which experts involved in the knowledge elicitation process provide their knowledge in the same domain.

User: Anyone with an objective of exploration a difficult-to-access environment. The user may or may not have expertise in the process or the methodology of exploration.

Expert: Selected individuals for knowledge elicitation process to include their knowledge in the Phoenix Knowledge Base. The experts involved into the knowledge elicitation process in Phoenix are shown in 4.2.

Knowledge Engineer: Experts involved in the process of building and maintaining knowledge bases.

Exploration goal: The knowledge about an unknown environment that the user wants to gain. It is identified from the user interaction starting from the user question.

Hypothesis: Proposal for a starting point of investigating an exploration goal where there is no or limited amount of evidence about its truth.

Experiment: The process of performing co-evolutionary and/or reincarnation cycles.

Exploration task/process: the process of performing experiments to answer the exploration goal. It can be decomposed into smaller exploration tasks/processes. An exploration task/process can be iterative where experiments are performed multiple times.

4.2 Knowledge engineers and domain experts

Table 7 shows knowledge engineers that will perform the knowledge elicitation and representation tasks. Table 8 lists the general knowledge domains that are involved into the Phoenix framework, and corresponding experts and their affiliations. Listed knowledge domains will be further specified based on the project requirements; therefore, the expert list will be extended accordingly during the knowledge elicitation process in WP3 (e.g. expertise on the different application environments and their dynamics, and components of the physical exploration devices).

Table 7. Knowledge engineers who contribute to the Phoenix Knowledge Base development process.

Tasks in WP3	Name	Affiliation
Knowledge Elicitation, Knowledge Representation, Technological Choices	Anil Yaman, Dr. Matt Coler, Dr. Giovanni Iacca	INnovation Center for Advanced Sensors and Sensor Systems (INCAS^3)

Table 8. Domain experts contribute to the knowledge content of the Phoenix Knowledge Base.

ID	Domain	Type of Knowledge	Expert
1	- Exploration procedures (Flume, Oil Reservoir) - Reincarnation cycle - Artificial and natural environments	Domain, inference and procedural knowledge	Dr. Rer. Nat. Elena Talnishnikh[1]
2	- Exploration procedures (Flume) - Localization and post-processing	Domain, inference and procedural knowledge	Erik Duisterwinkel[1,4]

3	- Exploration procedures (pipeline) - Artificial environments in controlled lab settings (pipeline, t-junction) - Virtual Environment Models	Domain, procedural and inference knowledge	Dr. Ir. Dirkjan Krijnders[1]
4	- Exploration procedures (flume)	Procedural and inference knowledge	Prof. Dr. Heinrich Wörtche[1,4]
5	- Localization and post-processing	Domain knowledge	Stephan Schlupkothen[3]
6	- Accelorometer	Domain knowledge	Dr. Martin Andraud[4]
7	- Co-evolutionary cycle	Domain Knowledge	Dr. Giovanni Iacca[1] Ahmed Hallawa[3] Anil Yaman[1,4]

***Affiliations:** [1]INnovation Center for Advanced Sensors and Sensor Systems (INCAS[3]), [2]Katholieke Universiteit Leuven (KU Leuven), [3]Rheinisch-Westfaelische Technische Hochschule Aachen (RWTH), [4]Technische Universiteit Eindhoven (TU/e)

4.3 Knowledge elicitation methods

Interviews

Interviews can be unstructured, semi-structured, or structured (a distinction addressed shortly). As an expert's facial expressions and gestures can reveal insight into his/her insight-making processes, video recording can be a useful way to ensure a clear interpretation of potentially subtle pragmatic indicators. Interviews should be recorded with a suitable digital recorder.

Unstructured interviews are those in which the elicitor and expert merely converse in general about a domain can yield some information. Such interviews are notoriously complicated insofar as the expert can get easily side-tracked or may make assumptions about the degree of expertise of the elicitor. Furthermore, unstructured interviews are exceptionally difficult and time-consuming to analyze. Nonetheless, especially in cases where the knowledge of the elicitor is extremely limited, an unstructured interview (perhaps based on material gathered in a list) may be a good place to start.

Semi-structured interview questions usually require greater explanation than simple descriptive questions, and thus refer to the structure of the expert's knowledge. Such questions include those like "Are there different kinds of X?", "Are there different ways to use X?" "How does one know when an X loses quality Y?" The semi-structured interview is similar to the structured one, but allows for a greater degree of flexibility in answers and permits the interviewer to interject spontaneous questions to follow up on potentially interesting leads — however, similar to unstructured interviews, the cost is that the analysis is generally more complicated.

Structured interviews are comparatively easier to administer and analyze than unstructured ones. They typically involve descriptive questions, moving from general to specific. In standard practice, they focus on the intersection of nine categories: space, object, act. activity, event, time, actor, goal and feeling. It is also common to ask specific probe questions to get at the structure of knowledge — see, for example, (Shadbolt & Smart, 2015). Other standard methods include the use of contrast questions. Contrast questions employ some of the basic ideas of Levi-Strauss and Saussure and applies them to fieldwork methods. The idea, then, is to uncover the meaning of a symbol by uncovering how it contrasts with others in the same domain. The result, at least in some cases, are emic scales in which terms are ranked along a domain; that is, they come from experts' categories and are not responses to scales or categories defined by the elicitor. Elicitors may also use test cases (Grover, 1983), a "first-pass knowledge base" (a meaningfully-organized list of propositions that express core concepts, terms, and rules in the domain — the expert goes over this and makes comments (Hoffman, 1987)) and groups of multiple experts to add structure to interviews. In some structured interviews, the elicitor may prepare "domain specific probe questions". These questions should cover a wide range of issues within the domain and be worded to avoid bias or ambiguity. The structure of the interviews can be made in-situ by an expert him/herself. This is done by simply asking the expert performing a task in the domain what s/he is doing and listing the responses. The elicitor follows up every single item on the list with a further question: "What kinds of questions will it make sense for me to ask about X?" These questions, in turn can be used to pursue each topic further with that expert or with others. To get at subdomain knowledge, experts may be asked to solve problems (provided by other experts) with incomplete information or talk through tasks which are provided with only limited information, targeted specifically to fill in gaps. In an effort to get at the more subtle aspects of the expert's knowledge, the elicitor should endeavor to present the expert with a tough case — that is, one with some specific challenging features. Such cases can be thought up by fellow experts or even by the expert him/herself. Finally, a participatory technique includes games like "20 Questions" may also be employed in situations where there are two experts, one interviewing the other (Cordingley, 1989). Alternatively, during the course of a structured interview, the elicitor may use generic questions designed to take the expert away from the domain of expertise in a certain respect. A typical question like "What's the difference between X and Y?" and "Can you give me more examples of X" (where X and Y are known from prior interviews/observations) can help the elicitor validate and refine the knowledge. The elicitor should take extra care to avoid reductive bias, overly simplifying a domain concept made up of terms that overlap with those same words (or translation of words) in everyday use.

Observation

Participant observation and task analysis are included as a necessary component of this methodology in light of the fact that not all knowledge is encoded linguistically (e.g. implicit knowledge and know-how); much is learned through observation of one kind or another and is thus evident only in behavior of experts. Special attention should be extended to exploring the indexical nature of the relation between things perceived by sensory experience and humans. The purpose of these components is to identify different tasks and subtasks in a given (typical/atypical) activity. This can be accomplished in a variety of ways, but the best place to begin is with a thorough study of experts performing task requiring expertise.

Once the elicitor believes s/he has a sufficient understanding of the experts' approach, the elicitor should interrupt and ask questions and/or perform structured interviews in-situ. Some researchers have established "basic types" of tasks (diagnosing, planning, designing, explaining). In any case, these tasks typically involve the following factors (Hoffman, 1987):

1. Analysis of complex stimuli into relevant features or cues based on "perceptual learning"
2. The analysis of conceptual categories in terms of relevant features (i.e. the perception of similarities and differences)
3. The analysis of the features and categories in terms of relevant underlying causal laws ("concept formation processes")
4. Abilities to infer and test these hypotheses

However, this in itself may be insufficient in the sense that it may not yield insight into the expert's reasoning processes. At least it should lend the elicitor a general overview of aspects of the field relevant to take into consideration.

Thinking aloud problem-solving

Also possible here are "think aloud" techniques in which the experts are instructed to generate a protocol. This method has a solid history in research on medical diagnostics and in many other areas. For this to be maximally useful, it may be necessary to train the expert to think aloud — a task that is easier said than done as articulating one's thoughts can impede trains of thought for some. Once this skill is acquired by the expert, it can be employed to other ends, for example providing scenarios that require intense processing but limiting the expert's time to solve it, etc.

In any case, these techniques should be analyzed by the elicitor and experts and broken down into representative procedural/behavioral chunks for the corpus. This can be done with reference to fieldnotes and photographs along with the explanations of the experts themselves.

Card sorting

Images/names of labeled items/events extracted from the corpus are provided on separate pieces of papers and informants group them together. Interpretation may be problematic because the groupings cannot be said necessarily to have employed any culturally relevant principles. Multiple sorts can be correct. More complex approaches are also possible, for example, sorting of photographs of a specific animal or plant can be compared with sorting attributes/characteristics of such animals/plants.

Triadic elicitation

A variety of forced-choice sorting in which the expert is shown three cards at a time and must pick the two that are the most similar, eliminating the one which is least similar. Once the choice is made, the researcher should attempt to understand the motivation. This can be used to elicit componential analysis data as well as material for an improved corpus. Variations include activities in which experts provide a list of some items as well as dimensions (arising from the triad sorting) on which these items can be

compared. That is, experts choose three members two of whom are similar on some dimension, the result can be viewed as a repertory grid (Geiwitz, J and Kornell, J and McCloskey, 1990).