Unveiling the implicit knowledge, one scenario at a time

Flavien Lécuyer, Valérie Gouranton, Aurélien Lamercerie, Adrien Reuzeau, Bruno Arnaldi, Benoît Caillaud

CGI2018 manuscript No.
(will be inserted by the editor)

Unveiling the implicit knowledge, one scenario at a time

Flavien Lécuyer [1] · **Valérie Gouranton** [1] · **Aurélien Lamercerie** [2] · **Adrien Reuzeau** [1] · **Benoît Caillaud** [2] · **Bruno Arnaldi** [1]
[1]**Univ Rennes, INSA Rennes, Inria, CNRS, IRISA**
[2]**Univ Rennes, Inria, CNRS, IRISA**

Abstract When defining virtual reality applications with complex procedures, such as medical operations or mechanical assembly or maintenance procedures, the complexity and the variability of the procedures makes the definition of the scenario difficult and time-consuming. Indeed, the variability complicates the definition of the scenario by the experts, and its combinatories demands a comprehension effort for the developer, which is often out of reach. Additionally, the experts have a hard time explaining the procedures with a sufficient level of details, as they usually forget to mention some actions that are, in fact, important for the application.

To ease the creation of scenario, we propose a complete methodology, based on (1) an iterative process composed of: (2) the recording of actions in virtual reality to create sequences of actions, and (3) the use of mathematical tools that can generate a complete scenario from a few of those sequences, with (4) graphical visualization of the scenarios and complexity indicators. This process helps the expert to determine the sequences that must be recorded to obtain a scenario with the required variability.

1 Introduction

To translate complex procedures into scenarios for virtual reality (VR) applications, the variability of those procedures complicates the task for the developers. For instance, procedures with more than 1000 steps, and with 10 or more acceptable ways of completing the task are extremely difficult to translate into scenarios. Indeed, complex and variable procedures are difficult to formalize for the expert, because their explanations contain implicit details that must, in fact, be explicitly described for the developer. They are also difficult for the developer to create, because they take time to understand, and to write.

We propose a novel method for the creation of VR scenarios that aims to facilitate the expression of complex procedures. This methodology is based on an (1) incremental process including: (2) the recording in VR of an expert, as well as the use of (3) the use of mathematical tools to merge and generalize the observations to synthesize a complete scenario, capturing the complexity of the procedures. This complexity is also represented with (4) a graphic visualization and indicators.

In this paper, after the related works in Section 2, we describe the solution in Section 3, through the example of an automotive wheel replacement procedure, in which the order of the nuts to tighten is important. Finally, we illustrate the method with a use case developed in collaboration with medical staff for the formation of nurses, in Section 4. We also conducted a user study to evaluate our solution, which we present in Section 5.

2 Related works

The problem of creating scenarios for VR applications has been tackled multiple times with different kinds of solutions.

2.1 Scenario models in VR

Several works have been proposed to integrate scenarios in virtual environments. Since we focus on scenarios engines that can be used by people that are neophytes regarding coding, we here focus on scenarios engines

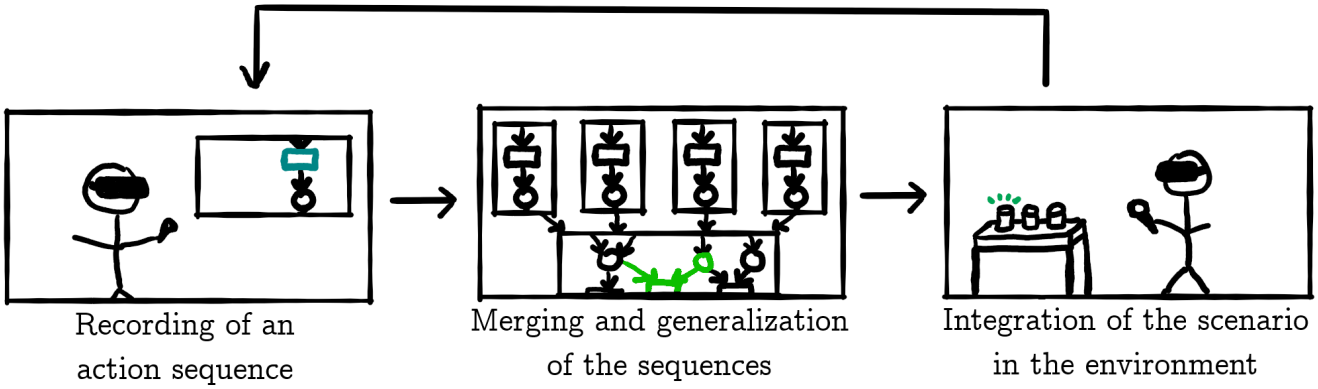

Fig. 1: Our method lets the expert iterate to create the scenario through several recordings

that integrate a visual representation of the scenarios. Although the abstraction brought by this visual representation might diminish the precision compared to coding [14], it helps the experts grasp the sequencing of the interactions, without having to look at lines of code of abstract concepts to understand it. There are several scenario models with visual representations. For each one of those models, the representation differs, according to the kind of formalism chosen by its authors, and the target use for the model. As an example, HAVE [9] is designed to be easily usable by developers. As such, its representation is based on UML notations, and more specifically on activity diagrams, which make the scenarios understandable as is by a developer. However, the experts are not always familiar with UML notations, and may need a phase of learning to be able to use those scenarios. Another well-known representation is used in LORA++ [13], with Grafcet-like scenarios. Although those scenarios can be legible for both the developer and the expert, the link between the scenario and the virtual environment is difficult to grasp with this representation. Because of this, it is difficult to fine-tune the scenario once a first version of it is obtained. To improve both the legibility of the representation and the connection between the scenario and the environment, some models are based on more abstract concepts, which they specialize to make them usable with VR. One of those models is HPTS++ [17]. In this model, the scenario is represented as a finite state machine, with the use of scripts complementing this scenario to connect it to the virtual environment [11]. Another use of finite state machines is proposed with Story Nets [6], which integrates more closely the code, although the state machine representation in itself is not enough to represent all the concepts needed for the scenarization of virtual environments: those additional concepts are represented in the code. A possibility to represent more visually those additional concepts is to derive the original graph representation to make those concepts clearer. In IVE [6] for instance, two concepts, the actor and the preconditions, are added to the visual representation to make it more expressive. This is done by replacing the notion of places that can be found in Petri nets with something that can manage both the actor and the preconditions. Another manner consists in extending the original concept. This solution can be found in ABL [21] which extends finite state machines for the definition of virtual agents behaviours. For a more general use of scenarios, #SEVEN [10] extends Petri nets to define the sequencing of the interactions in the environment.

2.2 Process mining

An interesting method for the creation of scenarios is Business Process Modeling (BPM), as its goal is to generate sequencing models, that can be seen as scenarios, through a set of observations. The BPMN language[1] is an industry-standard domain specific language for modeling business processes, that can be used to express and analyze the workflow of an industrial or administrative process, in enterprise information systems, for instance. The semantics of the language is quite reminiscent of Petri nets and the mapping of BPMN models to Petri nets is a simple syntactic transformation [12]. Although process mining allows to define scenarios from observations, the goal of process mining is to obtain compact scenarios from large datasets. Creating those datasets would require an important amount of time for the expert, which makes this solution not applicable in our context.

Scenarios can also be generated through Petri Net synthesis methods [4]. The synthesis method is a process mining techniques [1], with the striking difference

[1] `https://www.omg.org/spec/BPMN/2.0/About-BPMN/`

that scenario synthesis aims at enabling a variety of possible behavior from few recordings, while process mining methods follow the opposite aim of synthesizing concise models from large datasets. A particular class of nets, called Test and Flip (TF) [7], are used to model scenarios. Using only a few recordings, the idea is to generate a TF net representing the recordings given as input. TF nets have been introduced as a mild extension of elementary nets systems (ENS) [3] and Flip-Flop nets (FF) [23]. More details about test and flip synthesis is given in Section 3.4.

2.3 Recording in VR

While all those models can help the expert understand the scenario, it is still difficult to use them to author the said scenarios. Indeed, the expert still needs to learn the scenario model to be able to create scenarios. Usually, this learning includes some part of computer science learning, which makes the models hardly accessible. An accessible and intuitive way to author content, including scenarios, is the concept of creating by doing. This concept is based on the transformation of a demonstration into the content sought by the author. An interesting example of this can be found in the work of Angros et al. [2], where the user is recorded to generate a first version of the scenario. This first version is then semi-automatically generalized through a simplification process, which uses the objectives expressed by the user to generate a possible scenario respecting the constraints inherent to the objective. Although this method is useful for the definition of short scenarios, longer scenarios can only be defined by recording each atomic procedure separately and connecting them afterwards, which makes it more troublesome for the expert. The recording of a user, during a performance in VR, has been reused in different ways to provide a form of replay of the actions, whether it is a movie replay or a simple trace of the performance. An obvious use of the replay is to capture the user's movements and to replay them as a movie. This has an interesting use for the learning of dance moves, as presented in [8]. While this kind of replay can be useful for this use case, it is done by recording the movement as a continuous data, without any segmentation. A similar effort can be found in the work of Bailenson et al. [5], a solution to learn martial arts in which the users can go back in time to visualize themselves. A major drawback of this approach is that the user needs to wear clothes designed for motion capture. Because of this, the methodology is hardly exploitable for a generic tool. Two works that use the recording in an interesting way are *Mystery at the Museum* (*M@M*) [15] and *Environmental Detectives* [16]. In those works, the actions performed by a user can be replayed as a means to help the players remember what they did. Unfortunately, their approach lacks details, and the replay feature seems to be limited, and presented more as a textual log than a scenario that could be re-injected in the environment. Another form for the reused recording can be an animated movie, as proposed in *Heroes, Villains and Magicians* [22]. In this work, a story is influenced by the choices made by the users. The choices are logged and then reused to generate a movie, where the choices of the user are respected. Although this reuse of the recording is quite advanced, the movie is still a passive medium, and cannot really be re-injected in the environment to create a scenario.

2.4 Analysis

To conclude on the related works, we can notice that, although the question of simplifying the creation of scenarios for VR is not new, authoring a scenario is still hardly accessible for an expert. The main improvements in the creation of scenarios have been done by proposing scenario models that create a layer of abstraction. While beneficial, this layer is still not enough for the expert to become the author of the content. To facilitate the definition of those scenarios, process mining techniques can be used, but a synthesis oriented method is more efficient to manage the variability of the scenario. Indeed, synthesis methods are better fit to generate scenarios with few observations. Moreover, while planification methods can also help, they need the expert to be able to formalize the problem, which is often not possible without help, and thus inadequate for our needs, especially for implicit knowledge which is difficult to express. In parallel, some works focused on an approach of creation through actions to create content through the recording of a user. Unfortunately, none of those works generates a scenario that could be re-injected in a VR application.

3 Method

To ease the creation of complex scenarios, we propose a novel method, combining the automatic transformation of the user's actions into a scenario, and the use of a mathematical tool to help the expert generalize the obtained result. The method is separated into 4 steps, illustrated in Figure 1: the recording of an expert's actions to create a sequence, the merging and generalization of the recorded sequences, the integration of the obtained scenario in the environment, and the iteration over the previous steps. First, the expert's actions are

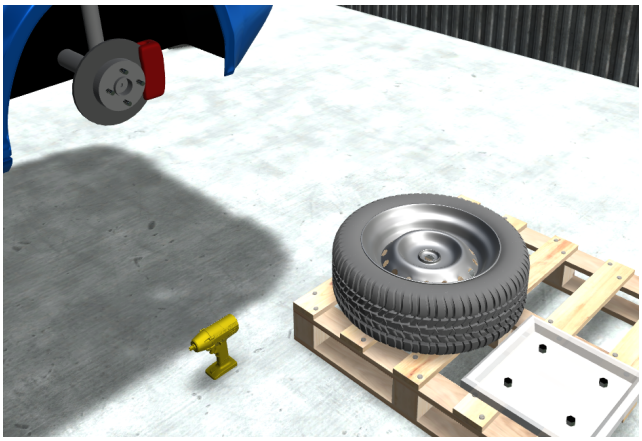

Fig. 2: The wheel and tools for the "changing wheel" scenario

Fig. 3: The pattern to follow (it can be symmetrically reversed, and started from any nut)

recorded to create a sequence, which can be used as input for the generation. This part is presented in Section 3.2. During this recording, some feedback is given to the expert as a mean to monitor the progression, as described in Section 3.3. Then, the mathematical models presented in Section 3.4 are used to merge and generalize the sequences, by inferring rules to generate sequences that have not been observed before. Once the new scenario has been generated, it is integrated in the VR application, as presented in Section 3.5. To illustrate the approach, we present how it can be used with the very simple example of changing a wheel. This example is presented in Section 3.1.

3.1 Changing a wheel

When changing a wheel, it is important to respect a cross pattern when tightening the four screws, such as the one in Figure 3. Indeed, doing otherwise puts the wheel at risk of bending, because too much pressure would be put on one side of the wheel. Out of 24 possibilities (4 nuts, hence 4! = 24 orders) for the order of the nuts, that makes only 8 that are in fact acceptable: in the other possibilities the cross pattern is not respected. While the pattern is known by the experts, it is difficult for them to write the scenario directly with a scenario model. Although constraining the pattern with planification would be possible, defining the goals is a bit complicated: the final goal is to have all four nuts tightened, but there is also an intermediary goal in which two opposite nuts must be tightened. Because of this, it is too complicated, even with this small example, for the expert to define either the constraints for a planification tool or a scenario with a scenario model.

Fig. 4: The expert manipulating the objects, with the corresponding view on the left

The user starts with a new wheel, the car on which the old wheel has been removed, a set of nuts to hold the wheel, as in Figure 2. To obtain the complete scenario, the expert needs to perform multiple variations of the procedure directly in the virtual environment, with as much variability as possible, so that the generation algorithm gets enough input data. To help the expert in finding which sequences must be performed to complete as much as possible the scenario, the highlights show the nuts that were tightened in the previous runs. In addition to this, indicators such as the number of new observed and generated variations are displayed, to indicate how efficient the generation was, along with the visualization of the scenario that was obtained. Thanks to this, the expert is able to vary the sequences and give more data to the generation algorithm. This provides an important time gain for the creation of the scenario, as well as the certainty that the scenario obtained this way is exempt of any mistake that would occur when writing it by hand.

3.2 Recording

The first step of the scenario authoring is the recording of an expert's actions, directly in the virtual environment, as in Figure 4. The goal of the recording is to translate the actions of the user into a sequence.

In this work, we represent the scenarios, as well as the sequences, with Petri nets, in which the tokens in

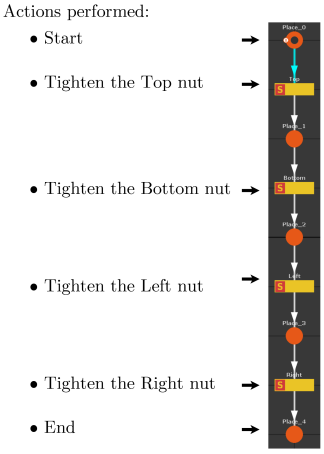

Fig. 5: The translation of the actions into a sequence

the places represent the state of the virtual world, and the transitions are triggered by the actions performed by a user in the environment. The choice of Petri nets is motivated by the compatibility with the generation algorithm presented in Section 3.4.

The recording is done by detecting the actions done in the virtual environment, and adding them to a sequence. At the beginning of the recording, the scenario is only a single place. Each time the user executes an action in the virtual environment, it is added to the scenario under the form of a new transition and a new place.

By doing this, the actions performed by the expert are translated into a sequence containing all those actions, chained in order. This sequence has the appreciable property that it can be used as input for the generation, since the translation of the actions into transitions is managed by the scenario model itself. This method of translating each action to generate a linear sequence is based on the approach presented in [19].

Although the translation of atomic actions is easier, it is possible to capture continuous actions, as long as the conditions for the beginning and the end – and, if needed, intermediary states – can be defined. Furthermore, we base the interactions used in the scenario on the object relation concept [20], which allows an efficient connection with our scenario model, as well as the preparation of the interactions prior to the recording. Indeed, the object relation paradigm is strongly based on the definition of abilities on the objects, and possible interactions in the environment, which is compatible with our approach. Another advantage of the object relation paradigm is that the interactions can be defined by the experts themsleves, through visual programming approaches such as the one in [18].

For the example of the cross mounting of a spare wheel, the expert will perform four actions during the recording, corresponding to the four nuts to tighten. As shown in Figure 5, this will automatically be translated in a sequence, with the correct formalism.

Another advantage of this method is that the expert does not need to formalize their knowledge to be able to describe a scenario. Indeed, the recording in itself will make this knowledge explicit, by forcing the expert to perform the different action. Thanks to this, each action that would normally be ignored, because it is considered as obvious for the expert or because the formalism would not represent it, is in fact recorded and taken into account in the final scenario.

Then, the generation tool is called with all the sequences recorded. The scenario obtained through the generation is then re-injected in the environment, to let the expert see how much of the variability is managed by the current version of the scenario. This re-injection is accompanied by a reset of the environment to the state it was in before the recording of the sequence.

3.3 Feedback

To help the expert evaluate the completeness of the scenario, two separate types of feedback are provided: an in-game scenario interface, and highlights derived from the recorded sequences.

3.3.1 Scenario visualization

The first feedback is done by having a scenario visualization interface directly in the virtual environment, as shown in Figure 6. The interface follows the user in the virtual environment, and can be anchored at a given point in space for more convenience. This visualization offers three main features: the visualization of the sequence being recorded and the generated scenarios, the controls for the generation process, and the creation of sub-scenarios, to ease the segmentation of the whole final scenario.

The in-game scenario visualization is used to show the creation of the scenario in real time to the user, and to provide controls for the recording of the sequences, and the generation of new variations. This scenario displayed in the virtual environment helps the expert to see how furnished the graph is, and therefore to get a first idea of its state of completion.

To let the expert to control the recording of the scenario, the visualization features interactive controls. Through this interface, the expert is also able to start and stop the recording of a sequence. At any moment during the recording, the expert can start what is called

Fig. 6: The main view of the scenario visualization in the virtual environment

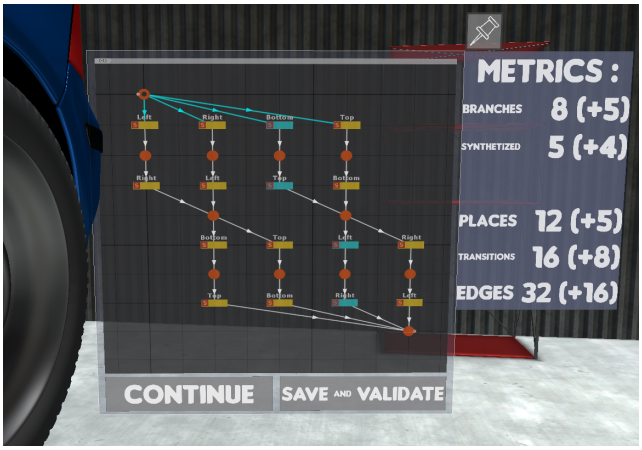

Fig. 7: The graphic visualization with the metrics associated

a "generation task". A generation task consists in recording several sequences to generate a scenario that takes into account the sequences, and generalizes them. To let the expert know how efficient the generation was, the total number of variations, as well as the number of new variations is displayed next to the resulting scenario, as shown in Figure 7.

When the scenario presents the need for variability, the expert can start by recording a simple scenario, and start a sub-scenario (or task), which will make use of the generation for more variability. This variability consists of having different ways to attain the same objective, with the same starting point. As such, it is extremely useful for procedures where some tasks must be in a certain order, but others do not for instance, and can be performed in any order, or even interleaved. Once that sub-scenario is completely recorded, the expert can go back to the enclosing scenario to continue the recording.

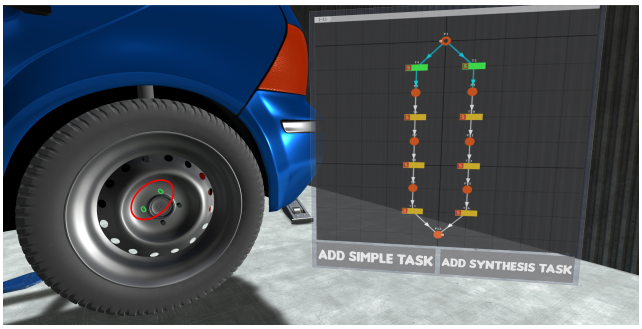

Fig. 8: The currently known paths are displayed to the user, along with the highlighted objects

3.3.2 Highlight of the variations

Another way to help the expert detect what paths are lacking in the synthesized scenario, is to highlight the actions already taken into account. Those highlights, placed on the objects, give a visual indication to show which actions already integrated in the obtained scenario. Thanks to this, it is easy for the expert to detect the actions that are missing from the scenario during the authoring. The generation is an incremental process, meaning that, at any point between the end of a sequence and the beginning of the next, the expert can either stop the generation if the obtained result is deemed exhaustive enough, or continue with another sequence a possible variation has not been yet added by the generation.

For the changing of the wheel, once the expert has recorded some sequences, the elements to tighten can be highlighted according to the paths that have been observed and inferred. An example of this is provided in Figure 8, where both the top element and the left one are highlighted to remind the expert that, in the recorded sequences, one of them was used as the starting point. The corresponding transitions in the scenario are also highlighted to help the expert make the correspondence between the scenario and the environment.

3.4 Merging and Generalization

A more complete scenario is generated once several sequences have been recorded in the environment. An example of the generation result is presented in Figure 9. The proposed approach falls in the field of process mining techniques. The principle applied is to generate an automaton representing the sequences given as input dataset, relying on a particular structure named Test and Flip net (TF net). This technique is able to generalise possible transitions that are absent from the original data and it also summarises the variability found

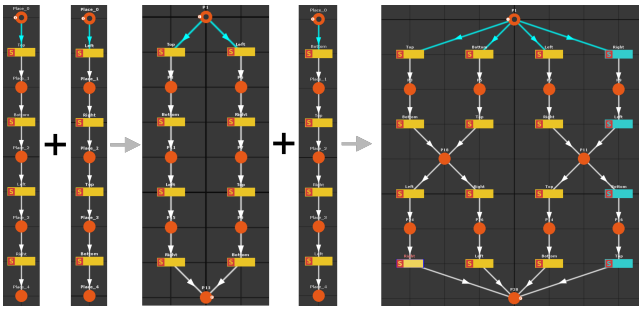

Fig. 9: An example of scenario generation, where 3 observations result in 8 possible paths (T=Top, B=Bottom, L=Left, R=Right). The transitions and paths in blue are generated from the observations in orange

across instances. While a traditional process mining usually relies on very large set of instances, the generation of a scenario with TF net synthesis must deal with a much smaller set, which may not include all the possible variations. Caillaud [7] introduced an initial version that we have adapted here for effective application in a VR environment.

A Petri net synthesis method, based on the theory of regions [3], is used. It consists of the generation of a particular class of 1-safe nets (each place can only bear 1 token at once), namely Test and Flip nets, which is a more efficient representation for the synthesis.

Intuitively, a TF net does not represent the scenario in itself, but the mechanics of activation and deactivation of possible actions in the scenario. This net can be associated with its marking graph, which represents all possible executions, and therefore more specifically the expected scenario. Such a graph also defines a language.

TF nets are therefore a very dense representation. They express concurrency, causality or conflict by construction. The synthesis algorithm consists in finding a certain TF net such that its marking graph define the least language of TF nets containing initial examples. All formal definitions used for formalization are given in the paper introducing the synthesis algorithm [7].

The integration of this technique with scenarios usable in VR requires some adaptations. Initially, the synthesis algorithm is used to generate a TF net. On the contrary, here, only the marking graph of the TF net is exploited that is an extension of the transition system obtained by the product of initial examples.

Overall, the adapted synthesis unfolds as follows:

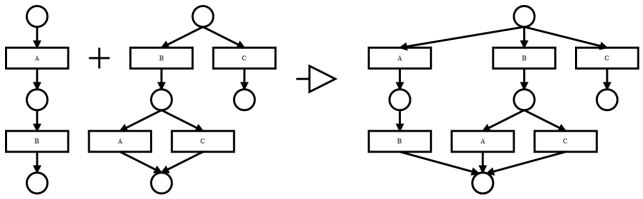

Fig. 10: The creation of the input scenario

1. Computation of a tree-like transition system, reproducing exactly the input scenarios.
2. Folding of this tree into a transition system, that may have cycle, but that matches the observation power of TF nets. This means that two sequences that can not be distinguished by any TF net well reach the same state.
3. Region-based synthesis [3] of a TF net, using a fast and scalable linear algebraic method [7].
4. Syntactic translation of the resulting graph into a scenario Petri-net.

The rest of this section provides details on these different stages.

3.4.1 From Petri Net to Transition System

The input scenarios are translated into sequence trees and then unified. By this way, the equivalent states are merged. We thus obtain some minor generalizations. Note also that this is done by keeping all technical data associated with events scenarios. This data will be used later to reconstruct the complete scenario in the desired form. Figure 10 illustrates this process.

3.4.2 Transition System Generalization using TF Net Synthesis

The next step is to extend the transition system respecting the constraints of a model, namely TF net. Following definitions and principles specifies the methodology used. A TF net is a generalization of elementary net, with 6 types of flow arcs, permitting a complete orthogonality between the test of a place and the alteration of the marking of a place. The firing of a transition is twofold: first, the markings of a set of places are tested to 0 or 1, and then, the markings places are eventually complemented. This leads to 6 possibilities in the 1-place net shown figure 11.

More formally, a TF net can be defined as a tuple $N = (P, T, a, b, c, m_0)$, such that P and T are respectively sets of places and transitions, a, b, c are three mappings defining the effect of a transition on the places and m_0 is the initial marking. The mapping a, b, c can

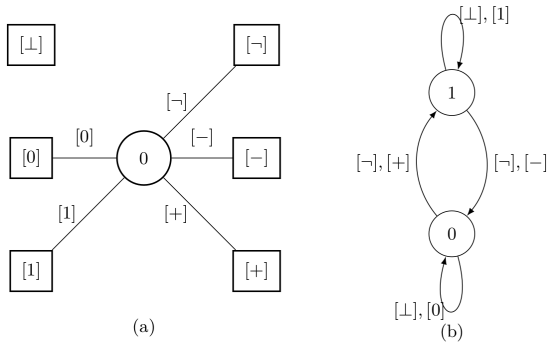

Fig. 11: From [7], one place TF net (a) and its marking graph (b). That graph defines the effect on a place of each type of flow arc (no link[⊥], complement a marked place [+], complement an unmarked place [−], complement a place [¬], needs a marked place [1], needs an unmarked marked place [0].

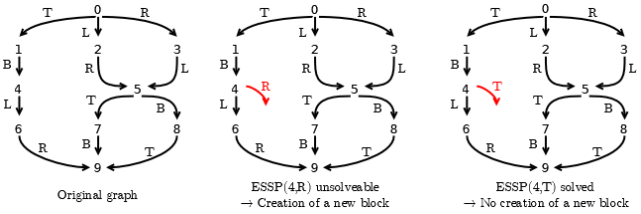

Fig. 12: The ESSP problem on the wheel example

also be characterized through three functions with 0 and 1 as possible values:

- *checks*(*place*, *transition*): does the transition checks the marking of the place?
- *marked*(*place*, *transition*): expected value of the place to trigger the transition. We can note that this function has a use only if checks is true
- *comp*(*place*, *transition*): does the transition complements the value of the place when triggered?

These functions somehow translate the TF net behavior. Moreover, a TF net can be associated with its marking graph. For reasons of efficiency, this graph is restricted to the markings and transitions that are reachable from the initial marking. This transition system is called the reachability graph of the TF net. The TF net synthesis algorithm computes a TF net which language of possible behaviors is the least language of a TF net that contains all the input scenarios. In this sense, the synthesis procedure is optimal, meaning that no TF net will allow fewer sequences of actions, while allowing all input scenarios. This algorithm is detailed in [7] and the mathematical of Region-based Petri-net synthesis can be found in [3].

Let us recall the essentials of the Theory of Regions, in the context of TF net synthesis. The main concept is that of *Event-State Separation Problems* (ESSP): given a state s and an action t, such that t is not allowed in s, the ESSP (s, t) consists in deciding whether there exists a place of a TF net, that would refuse to perform the action t in state s, while allowing all occurrences of t in the input scenarios. The TF net synthesis method consists in computing the places of the synthesized TF net, one at a time, by solving all ESSPs of the input scenarios. Therefore, each place of the synthesized net solves at least one event-state separation problem (s, t), meaning that the place forbids the firing of some action t in a state s. For instance, with the cross-mounting scenario in Figure 12, the ESSP is computed for each place, and for each action not already allowed for this place (e.g. for the state 0, only the `Bottom` action is missing, for state 4, there is `Top`, `Bottom` and `Right`). The solveability of the ESSP means that the transition should not be added; and complementarily its unsolveability leads to the creation of the transition.

The key result of [7] is that ESSPs can be encoded as a system of linear Boolean equations, that can be solved by classical XOR-Sat methods, based on Gaussian elimination [24]. This guarantees the scalability of the method, both in terms of the number actions, and of the number of states of the input scenarios. Our software implementation takes the different steps proposed in [7], with some simplifications and optimizations to improve performance. Namely, the last steps of the algorithm (steps 4 and 5 of the cited paper) are not required, since only the reachability graph of the synthesized net is used.

3.4.3 From Transition System to Petri Net

The transition system obtained can ultimately be transformed into a Petri net. Each place in the transition system corresponds to a place in the Petri net, while the relations between the places become transitions associated with an incoming arc and an outgoing arc from or towards the corresponding places. The technical data associated with the transitions is included.

3.5 Integration

Once the scenario has been generated through the sequences, it is integrated in the virtual environment to provide additional feedback to the user. Since the feedback consists in highlighting the elements that can be used to move the scenario forward, it can also be directly used as is for the final application. However, the experts may wish for other possibilities of integration

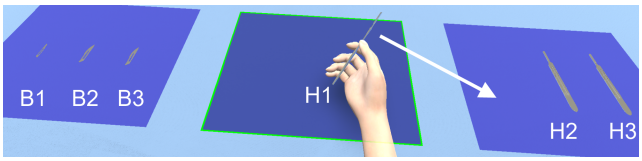

Fig. 13: The operating table

of the scenario. To facilitate this, we provide tools to reuse the scenario in different ways, to allow a reuse adapted to the needs of each application.

The main possibility for reusing the created scenario is to re-inject it as a guide for a user. Since the scenario is created with the help of the scenario engine used to read them, the scenario created is directly usable for this kind of use.

For more complex reuses of the scenario, for which it would be difficult for the expert to bring the fine grain modifications needed to either the scenario or the virtual environment, the intervention of a developer is beneficial.

To help the developer continue the project after the work is done by the expert, the scenario can easily be sent along with the virtual environment, under the form of a Unity (`http://unity3d.com/`) project. Thanks to a graphic editor we provide, the developer can easily modify the scenario afterwards.

To complete this modification, we designed tools to use the scenario in different ways in the virtual environment, either to guide the final user of the VR application, or to add virtual agents capable of performing parts of the scenario.

The final scenario provided for the final application is fully integrated within the virtual environment, but can also be read thanks to its graph format. In this form, the scenario obtained through the recording process can be extremely useful for the expert, as a support to study their work process. Indeed, a quick look at the scenario is enough to apprehend the complexity of the scenario. Analysing the scenario in more details also helps in understanding the process, and finding possible patterns that would not be detected otherwise.

4 Surgery training

The second use case we designed concerns the preparation of an operating table. The preparation of the operating table is particularly difficult to learn, since there are many objects to prepare, and each operation needs a different preparation.

For the use case, we focus on the assembly of three scalpels, as shown in Figure 13. As shown in Figure 14, there are 80 possible variations of this scenario, deriving

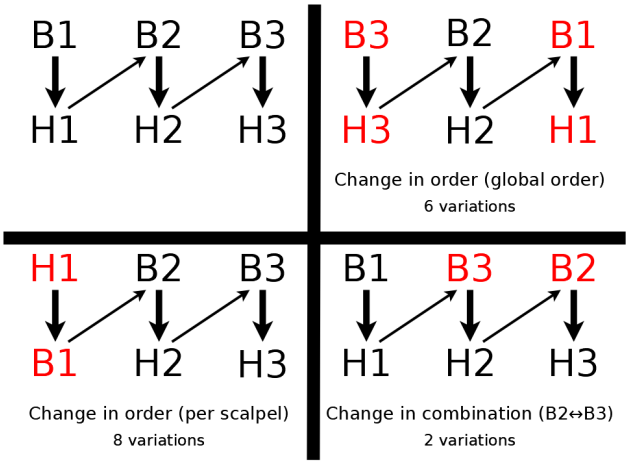

Fig. 14: Multiple variations (the changes are displayed in red) can impact the combinatory for the scalpel assembly. Combined, those variables give 6×8×2=92 possibilities, out of which 80 are unique.

from 3 variables. Indeed, the scalpels can be assembled in any order, making $3! = 6$ variations. Since two of the scalpels (the second and the third from the left) have the same handle and blade, two blades can go for the same handle (and vice-versa), multiplying the number of variations by 2. Finally, for each scalpel, the user can take either the handle or the blade first, giving $2^3 = 8$ cases. In total, this gives $6 \times 2 \times 8 = 92$ possibilities. Once the doubles have been simplified, this leaves out 80 cases. It should also be noted that the scalpels are just a part of the complete intervention, which consists of preparing the whole table. Because of this, and although the scenario may seem simple at first, it is in fact quite difficult to model, as there is a lot of variability to take into account (does the user take the blade or the handle first, which blade goes with which handle, which scalpel is assembled first, ...), making the 80 possibilities in total in the complete scenario. It is to note that this number does not take into account the hand used to take the objects, and hence is less important than what it could be in that particular case. Although the scenario can be legible, and could be written by a developer given enough time, an important challenge here is to write it without introducing any mistake, which is tedious given the number of actions to encode. Writing this scenario proves quite difficult, and hence requires a lot of time and generates errors easily. However, it takes into account whether the blade or the handle is taken first, as it may matter for the sake of sanitizing. To illustrate this complexity, the scenario for this use case is provided in Figure 15.

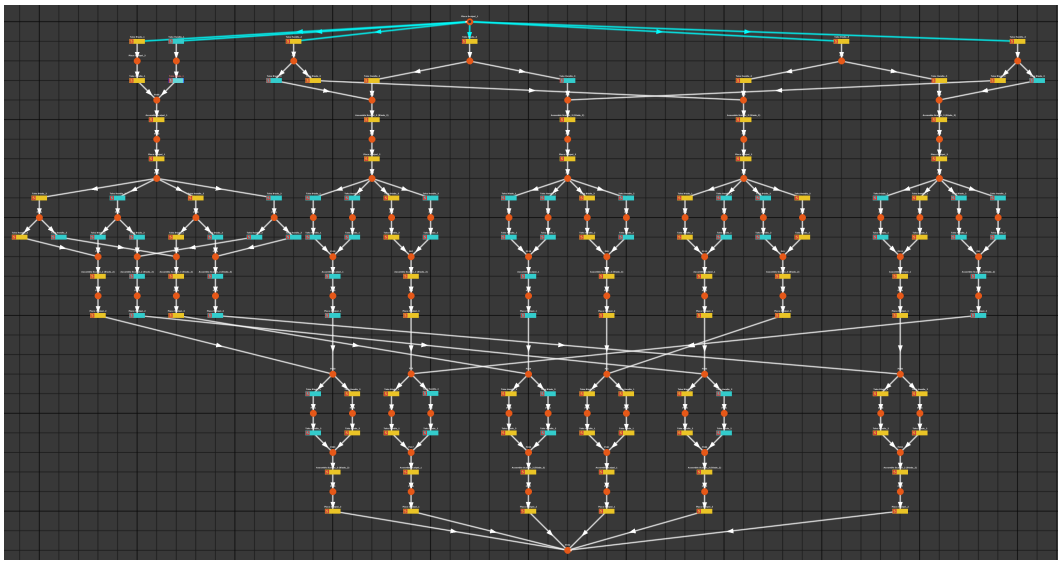

Fig. 15: The synthesized scenario for the scalpel assembly, from 8 observations (the generated transitions are in cyan)

For this scenario, the main advantage of the generation is that it saves a lot of time to write the scenario. Indeed, while it would take a few hours, even for the developer, to write the complete scenario (assuming it is written without any errors, and assuming that the variability is perfectly known beforehand), it takes less than half an hour to record the needed sequences and get the final scenario. Indeed, eight sequences – if they represent all the variability that may stem from this problem – are enough to generate the whole scenario in Figure 15.

Furthermore, and in the same manner as the cross mounting use case, defining the constraints using only the state of the world is rather difficult. Although it would be possible to manage a part of the complexity of the problem by defining that a handle and a blade can be assembled only if they are of the same type, this does not completely solve the problem. Indeed, defining those constraints in the virtual environment requires to add some code to the possible interactions, which would mean that a developer is needed prior to the recording, and that the developer should already be aware of the constraints that are important to put in the virtual environment.

5 User study

In order to assess the benefits provided by our tool, we designed a user study, with a total number of 30 participants. The participants were divided into three groups, depending on their level of expertise with coding: 9 with no coding expertise, 14 had expertise in development, and 7 had expertise with the coding of scenarios in VR with the model used. After the explanation of the protocol and the consent form, we asked the participants to state how they would formalize a scenario for a use in virtual reality. In order to get a point of comparison with the method they would naturally use for the definition of scenarios, the participants were asked to propose a solution for the definition of the wheel changing scenario. After this, the proposed method of scenario authoring was introduced to the participants, and they were asked to fill a first questionnaire (previous experience in VR, previous experience in Unity 3D, SSQ).

Then, the participants were equipped with a HMD (a HTC Vive), and immersed for a training session of around 15 minutes, during which they could get familiar with the controls of the virtual environment, and create a first scenario with the proposed method. This first scenario was a simplified version of the scalpels scenario: with two scalpels, pick a blade and a handle, assemble the scalpel, and repeat with the second scalpel. The order to follow for the sequences was given to the participants, so that they could see the generation occur at least once. After this first scenario, the participants were free to continue with the first environment, until they felt ready for the second task.

The main task of the experiment was the creation of the wheel changing scenario with the proposed method. During this task, the participants where completely free to create the scenario how they wanted, without any help from the experimenter. Once they estimated that the obtained scenario was complete, they were free to

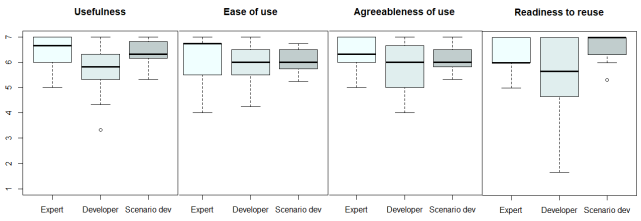

Fig. 16: Perceived usefulness, ease of use, agreeableness, and readiness for reusing the tool

validate and end the task. Out of the 30 participants, 2 obtained the wrong scenario (one did not generalize enough, and one other did not respect the cross pattern and obtained the 24 possible orders).

After this, the participants were asked to fill a second questionnaire (NASA-TLX, SSQ, SUS and UTAUT2 in this order). In total, the experimented lasted between half an hour (for the participants with a strong VR and coding expertise) to 75 minutes (for some of the participants with no coding expertise).

Overall, the results of the experiment show that the participants appreciated the proposed tool. The perceived usefulness, ease of use and agreeableness (from the UTAUT2 questionnaire) mainly vary between 5 and 7 (cf Figure 16). The highest scores for the UTAUT2 questionnaire were obtained with the experts and the scenario developer, who were overall really pleased by the tool. However, the regular developers were less inclined to find the tool useful enough to be reused: although they found it useful, they stated that it would take, in their opinion, more time to use the tool than to complete the proposed scenario with code. Also, the scores for the regular developers vary much more than the ones for the other two groups, especially for the questions asking whether they would be ready to reuse the tool or not.

6 Discussion and future works

We would like to highlight that this methodology does not completely replace the role of the developer in the process of creating a VR application. Indeed, there is still the need to model the virtual environment beforehand, and to add interactions to it. However, this methodology is able to help the experts easily create scenarios for those environments and use those in simple ways. This is a powerful tool as it allows them to take more place in the process, and helps both the expert and the developer to focus more on their respective strengths.

In future works, we plan to include other types of events that may trigger the transitions in the scenario, such as time constraints, or a single collision. In the same fashion, it would be interesting for the expert to be able to add specific consequences to the transitions, in order to customize more precisely the scenario obtained from the recording. Those other events would allow the expert to define more complex scenarios from the environment, with the possibility to change the pace and the behaviour of the scenario according to what the user is doing. This kind of modifications would be a powerful way to complement the control of the actions sequencing in the environment.

A second improvement for the method would be to take into account the wrong scenarios recorded by the user or generated. Labelling those scenarios would be a way to define counterexamples, and as such would help in preventing overfitting.

7 Conclusion

In this paper, we proposed a complete method for the creation of scenarios in VR, based on the recording of an expert's actions in VR, and the merging and generalization of the recorded sequences. To facilitate the creation of the scenario with this method, an iterative approach is proposed. With this approach, the expert can first record a sequence of action, use a mathematical tool to merge and generalize the sequences to create a scenario. This scenario is then integrated automatically in the virtual environment, to let the expert iterate over the first two steps with visual indications showing what is already managed by the scenario.

Thanks to this novel methodology, the experts are able to define scenarios for VR applications easily. Indeed, the recording helps them express their implicit knowledge by performing the actions instead of having to explain them, and the generation of new sequences from the observations helps them manage easily the variability of the procedures to scenarize. This methodology is an important tool to ease the communication between the domain experts and the developers, who often have difficulties when building VR applications that require both the knowledge of the expert and a certain amount of variability to get a scenario that is really pertinent.

8 Acknowledgements

This work is part of the ANR-16-FRQC-0004 INTROSPECT project, and the SUNSET project funded by the ANR-10-LABX-07-01 "Investing for the Future" program. We would also like to thank people from the

Hybrid team who provided relevant feedback on this work.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st edn. Springer Publishing Company, Incorporated (2011)
2. Angros Jr., R., Johnson, W.L., Rickel, J., Scholer, A.: Learning Domain Knowledge for Teaching Procedural Skills. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3, AAMAS '02, pp. 1372–1378. ACM, New York, NY, USA (2002)
3. Badouel, E., Bernardinello, L., Darondeau, P.: The synthesis problem for elementary net systems is np-complete. Theor. Comput. Sci. **186**(1-2), 107–134 (1997)
4. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer (2015)
5. Bailenson, J.N., Yee, N., Blascovich, J., Beall, A.C., Lundblad, N., Jin, M.: The Use of Immersive Virtual Reality in the Learning Sciences: Digital Transformations of Teachers, Students, and Social Context. Journal of the Learning Sciences **17**(1), 102–141 (2008)
6. Brom, C., Šisler, V., Holan, T.: Story Manager in 'Europe 2045' Uses Petri Nets. In: M. Cavazza, S. Donikian (eds.) Virtual Storytelling. Using Virtual Reality Technologies for Storytelling: 4th International Conference, ICVS 2007, Saint-Malo, France, December 5-7, 2007. Proceedings, pp. 38–50. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
7. Caillaud, B.: Surgical Process Mining with Test and Flip Net Synthesis. In: R. Bergenthum, J. Carmona (eds.) Application of Region Theory (ART), pp. 43–54. Barcelona, Spain (2013). URL `https://hal.inria.fr/hal-00872284`
8. Chan, J.C.P., Leung, H., Tang, J.K.T., Komura, T.: A Virtual Reality Dance Training System Using Motion Capture Technology. IEEE Transactions on Learning Technologies **4**(2), 187–195 (2011)
9. Chevaillier, P., Trinh, T.H., Barange, M., De Loor, P., Devillers, F., Soler, J., Querrec, R.: Semantic modeling of Virtual Environments using MASCARET. In: 2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), pp. 1–8 (2012)
10. Claude, G., Gouranton, V., Bouville Berthelot, R., Arnaldi, B.: Short Paper: #SEVEN, a Sensor Effector Based Scenarios Model for Driving Collaborative Virtual Environment. In: T. Nojima, D. Reiners, O. Staadt (eds.) ICAT-EGVE, International Conference on Artificial Reality and Telexistence, Eurographics Symposium on Virtual Environments, pp. 1–4. Bremen, Germany (2014). URL `https://hal.archives-ouvertes.fr/hal-01086237`
11. Cremer, J., Kearney, J., Papelis, Y.: HCSM: a framework for behavior and scenario control in virtual environments. ACM Transactions on Modeling and Computer Simulation (TOMACS) **5**(3), 242–267 (1995)
12. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and analysis of bpmn process models using petri nets. Queensland University of Technology, Tech. Rep pp. 1–30 (2007)
13. Gerbaud, S., Mollet, N., Arnaldi, B.: Virtual Environments for Training: From Individual Learning to Collaboration with Humanoids, pp. 116–127. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
14. Green, T.R.G., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages & Computing **7**(2), 131–174 (1996)
15. Klopfer, E., Perry, J., Squire, K., Jan, M.F., Steinkuehler, C.: Mystery at the museum: a collaborative game for museum education. In: Proceedings of the 2005 Conference on Computer support for collaborative learning, pp. 316–320. International Society of the Learning Sciences (2005). URL `http://dl.acm.org/citation.cfm?id=1149293.1149334`
16. Klopfer, E., Squire, K.: Environmental Detectives–the development of an augmented reality platform for environmental simulations. Educational Technology Research and Development **56**, 203–228 (2007)
17. Lamarche, F., Donikian, S.: Automatic orchestration of behaviours through the management of resources and priority levels. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3, pp. 1309–1316. ACM (2002). DOI 10.1145/545056.545124. URL `https://doi.org/10.1145/545056.545124`
18. Lécuyer, F., Gouranton, V., Reuzeau, A., Gaugne, R., Arnaldi, B.: Authoring AR by AR: abstraction and libraries. In: Y. Kakehi, A. Hiyama (eds.) ICAT-EGVE 2019 (2019)
19. Lécuyer, F., Gouranton, V., Reuzeau, A., Gaugne, R., Arnaldi, B.: Create by doing – Action sequencing in VR. In: M. Gavrilova, J. Chang, N.M. Thalmann, E. Hitzer, H. Ishikawa (eds.) Advances in Computer Graphics, pp. 329–335. Springer International Publishing", Cham (2019). DOI 10.1007/978-3-030-22514-8_27
20. Lugrin, J.L., Cavazza, M.: Making sense of virtual environments: action representation, grounding and common sense. In: Proceedings of the 12th international conference on Intelligent user interfaces, pp. 225–234. ACM (2007). DOI 10.1145/1216295.1216336. URL `https://doi.org/10.1145/1216295.1216336`
21. Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE Intelligent Systems **17**(4), 39–47 (2002)
22. Paiva, A., Machado, I., Prada, R.: Heroes, Villians, Magicians, & Dramatis Personae in a Virtual Story Creation Environment. In: Proceedings of the 6th International Conference on Intelligent User Interfaces, IUI '01, pp. 129–136. ACM, New York, NY, USA (2001)
23. Schmitt, V.: Flip-flop nets. In: C. Puech, R. Reischuk (eds.) STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings, *Lecture Notes in Computer Science*, vol. 1046, pp. 517–528. Springer (1996)
24. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: O. Kullmann (ed.) Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings, *Lecture Notes in Computer Science*, vol. 5584, pp. 244–257. Springer (2009)