Simulation Vignette Generation from Answer Set Specifications

Dashley K. Rouwendal van Schijndel
University of Oslo, Department of Technology Systems
d.k.rouwendal@its.uio.no

Jo E. Hannay
Norwegian Computing Center, Department of Applied Research in Information Technology
jo.hannay@nr.no

Audun Stolpe
University of Oslo, Department of Technology Systems; Norwegian Computing Center, Department of Applied Research in Information Technology
audun.stolpe@nr.no

ABSTRACT

We investigate an approach that allows exercise managers to design simulations with an explicit focus on building skills, rather than having to focus on all the objects and interactions that a simulation must have. Exercise managers may design exercises at various levels of abstraction and always independently of how those sessions are implemented in simulations, while simulation components that implement the design are assembled and to some extent, automatically, behind the scenes. We outline (1) how Answer Set Programming can assist exercise managers in exercise planning and (2) how automated stage and content generation may be used to invoke appropriate simulation components to realize the design. For deliberate and recurrent training of decision-making skills, stages and content must vary to avoid familiarity (testing effects). We conclude by distilling a main research hypothesis that stipulates how (1) and (2) represent two modes of automated reasoning (so-called deductive versus abductive) and how that distinction clarifies the planning task.

Keywords

Exercise Management, Answer Set Programming, Mixed Reality Simulations, Vignette Generation.

INTRODUCTION

Simulation-based crisis response and management training has challenges concerning structured planning, execution and analysis (Hannay and Kikke 2019). Consequently, organizations often have low awareness levels of the effectiveness of training and of what exactly has been trained (Grunnan and Fridheim 2017; Pollestad and Steinnes 2012; Skarpaas and Kristiansen 2010). There are technological challenges in achieving the necessary interoprability between systems for delivering the training system (Durlach 2018; Tolk 2012; Edgren 2012).

Therefore, work is being conducted on an exercise management architecture (Figure 1).

The architecture consists of four components:

1. A front-end *exercise management system* (a web application for easy access on portable devices), where an exercise manager can
   - declare training objectives,
   - select which essential skills to be trained to reach those objectives,
   - design events to stimulate development of those essential skills,
   - design corresponding metrics for measuring essential skill performance in the events and
   - compose series of events (vignettes; see below), from the events.

   The design of events and composition to vignettes in a graphical work space.

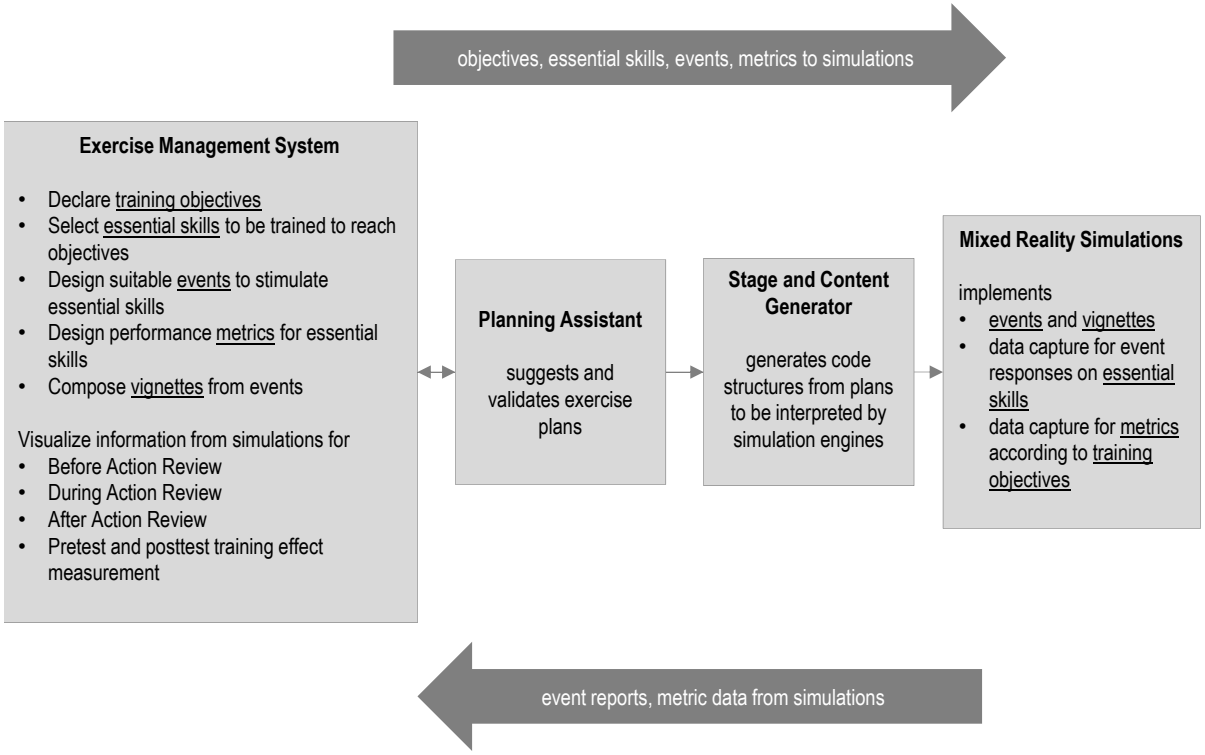

Figure 1. Exercise Management and Simulations (ExManSim) architecture

2. A back-end *planning assistant* based on Answer Set Programming (see below), which suggests and validates events and vignettes to ensure that the exercise manager composes events in a logically consistent manner in line with the essential skills to be trained.
3. A back-end *stage and content generator* that translates answer set programs to code structures to be interpreted by simulation engines, so that the exercise manager's design changes are reflected instantly in the simulations.
4. A suite of mixed reality simulation components that implement the designed vignettes, together with associated measurement of skill performance metrics.

This design promotes the idea of composing exercise and training vignettes from a selection of events that are designed to stimulate essential skills.

The front-end allows exercise management to visualize information for use in before-, during- and after-action reviews; in which exercise managers may assess and evaluate a training session at critical moments for validation of the consistency and coherency between objectives, skills, events and metrics, based on actual trainee performance. This gives explicit support to (Salas et al. 2009). Pretest and posttest effect measurements allow essential skill performance measurements in controlled simulations before and after training exercises.

The purpose of these functions is to support *deliberate practice* (Ericsson 2006b); a framework that addresses the short-comings of "learning on the job", by a strong focus on difficult aspects, immediate and tailored feedback (by a coach or computer-adaptive system), followed by tailored re-trials integrated into the larger sequence of tasks.

The mixed-reality simulations are an appropriate mix of real and synthetic actors, objects and events are composed in a shared reality. This gives the opportunity to use existing systems and equipment together with virtual elements in an optimized learning arena. Training's can be located at their normal work places. The collected metrics and other data will be utilized to analyse learning performance and to improve practices and procedures once in operational use.

Once a vignette has been composed in the front-end, the simulations for running the vignette must be composed quickly and accurately. Today, it typically takes months or even years to develop simulations; see e.g., (Edgren 2012). This is a serious disabler for generating simulations that match differing training objectives and skills to be stimulated through appropriate events in a flexible manner. Initiatives on *Modelling and Simulation as a Service* (MSaaS) work toward creating simulations for operations and training readily and rapidly (Hannay and van den Berg 2017; van den Berg et al. 2017; Asprusten and Hannay 2018) from simulation services. The *service*

concept embodies reusability by standardized common functionality, and composability through loose coupling and standardized service descriptions.

VIGNETTES AS SERVICES

The concept of MSaaS embodies several principles for how simulations can be constructed readily and rapidly for the task. The main focus is on offering simulation functionality in terms of shared reusable services with standardized interfaces which all systems in a distributed simulation agree to use. This also enables "fair fight" across systems, so users experience more or less the same things. Vegetation that offers cover for a vehicle in one simulation system must be represented and rendered in the other simulation system, effects of devices should be equal in all systems, etc.

So, while the focus of MSaaS so far is on extracting common general functionality, we shall focus on composing simulations from events and vignettes as services, rather than on simulation functionality as services. A *vignette* is

> *a reusable temporally ordered set of events and behaviors for a specific set of entities* (Simulation Interoperability Standards Organization 2018).

Offering events as services requires automation of the simulation building process. The aim is to find a method that takes a machine-readable specification and semi-automatically

- generates an appropriate *stage*; , an empty simulation environment which can be populated on the fly with content (objects and their relationships),
- populate the stage with content so that simulation-based training as specified in the exercise management front-end can be realized.

Stages must be designed to accommodate all content that vignettes must hold toward training the desired skills.

ESSENTIAL SKILLS

The present focus for ExManSim is on training decision making skills. There are several essential skills underlying decision making that arise from the fields of judgment and decision making (Gigerenzer and Todd 1999; Kahneman and Frederick 2004) and forecasting (Armstrong 2001). Here, we will focus on situation awareness (SA) (Endsley 2000), because it has particular relevance to crisis management; e.g., (Steen-Tveit and Jaziar 2019), and because it is readily defined in terms of the elements of simulations. SA is divided into three levels:

SA1: an actor is aware of the position of all relevant entities (things and people) in the situation

SA2: an actor is aware of the the relevant relationships between the entities

SA3: an actor is aware of the possible future configurations of the entities and relationships; thus understands how entities and relationships can evolve.

Example: bin fire in airplane hangar. Consider a simple training vignette "bin fire in airplane hangar", where the objective is to train employees appropriate decision making during a fire.

In the front-end, an exercise manager decides to train SA1 by specifying that the vignette should include the following entities: *oil-drenched paper* in a *bin*, an *initial fire*, one *fire extinguisher that is appropriate for the type of fire*, one *fire extinguisher that is unsuitable for the type of fire*, an *aircraft* in the immediate vicinity of the bin an *alarm button* and implicit *fire fighting resources*. The mode of visibility can also be specified; for example, for added difficulty, the fire extinguishers should be located behind a cubicle not in direct line of sight from where the bin is.

For SA2, the exercise manager might specify relevant relationships between these entities as follows: initial fire *can ignite* oil-drenched paper, appropriate fire extinguisher *can put out fire in* oil-drenched paper, unsuitable fire extinguisher *cannot put out fire in* oil-drenched paper, and alarm button *will call* fire fighting resources.

For SA3, the exercise manager might specify the future relationship, fire in oil-drenched paper *can spread to* aircraft.

Possible events are then defined based on relationships: The relationship between *initial fire* and *oil-drenched paper* gives rise to the event *fire in oil-drenched paper*. The relationship between *CO^2-based extinguisher* and *oil-drenched paper* gives rise to the event *fire gets extinguished*. The relationship between *water-based extinguisher* and *oil-drenched paper* gives rise to the event *fire does not get extinguished*. The relationship between *fire in oil-drenched paper* and *aircraft* gives rise to *aircraft catches fire*, etc.

The simulation would start with the event *fire in oil-drenched paper*. Performance on SA1 can be measured for example, by monitoring a trainee's time spent on detecting the fire and on moving towards where the fire extinguishers are situated. Performance on SA2 might be measured by monitoring whether the trainee chooses the appropriate fire extinguisher. If the trainee does not act correctly, the event *fire does not get extinguished* is simulated and a secondary fire event *aircraft catches fire* occurs. Performance on SA3 might be measured by recording whether the trainee presses the alarm button if the initial fire is not put out, thus realizing that the fire will spread to the aircraft and further resources are needed.

□

When generating the simulation for this simple vignette, the stage might be a large single room, with angles to place objects out of sight. The entities must then be placed according to specification and other constraints, such that the relationships are implemented and the events are enabled.

LEVELS OF ABSTRACTION FOR PLANNING

It is possible to plan a training session or exercise at various levels of abstraction. An exercise manager could be a person with a more general focus on employee health and safety, and with little domain specialized knowledge such as for fire fighting. Such a person might want to plan a training session or an exercise at the level of object types instead of specific objects.

Example: fire in indoor industrial space. The training vignette "bin fire in airplane hangar" can be abstracted to "fire in indoor industrial space" formulated as follows: To train SA1, *primary flammable material*, an *fire starter*, one *means of extinguishing fire that is appropriate for the type of fire*, one *means of extinguishing fire that is unsuitable for the type of fire*, a *secondary flammable material* to which a the initial fire may spread, a *means to call for professional help* and implicit *fire fighting resources*. The means of extinguishing fire should not be located in direct line of sight from where the primary flammable material is.

For SA2, the exercise manager might specify relevant relationships between these entities as follows: fire starter *can ignite* initial flammable material, appropriate means of extinguishing fire *can put out* fire in initial flammable material, unsuitable means of extinguishing fire *cannot put out* fire in initial flammable material, and means to call for professional help *will call* fire fighting resources.

For SA3, the exercise manager might specify the following future relationship: fire in initial flammable material *can spread to* secondary flammable material.

□

The above formulation of "fire in indoor industrial space" allows an exercise manager to focus on the salient relationships that give rise to events, irrespective of concrete items present in the event.

The abstract formulation can be instantiated to several concrete vignettes; for example to "bin fire in airplane hangar" but also to a vignette where, say, a fuel puddle or a water boiler catches fire initially and where a secondary fire in a fuel tank or a ventilation unit might ignite if the initial fire is not handled appropriately.

Levels of abstraction allow various level of exercise planning. Perhaps, a governmental exercise manager, in charge of overseeing national crisis training standards, designs exercises at the abstract level as templates to be refined by domain experts in various sectors. Also, various levels of abstraction enable efficient tool design and representation. Indeed, the vignettes we are discussing can be abstracted further. We said above that relationships between objects give rise to events which can be composed into vignettes. It is possible to declare abstract vignettes purely at the level of events, abstracting away from objects and relationships.

Example: crisis with two dependent events. An abstract training vignette "crisis with two dependent events" can be formulated as follows: The exercise manager might decide to train SA1, SA2 and SA3 by specifying that the vignette should include: An *initial event* with a *hazard* that can trigger an *initial crisis* targeted to train SA1 and SA2, with an *intended means to handle the crisis* and a *non-functional means to handle the crisis*, and a *secondary event* with a *hazard* that can trigger an *secondary crisis* targeted to train SA3, with an *intended means to handle the crisis* and a *non-functional means to handle the crisis*.

□

The abstract "crisis with two dependent events" vignette instantiates to more concrete vignettes in the examples above. It also instantiates to vignettes that are different from the "fire in indoor industrial space":

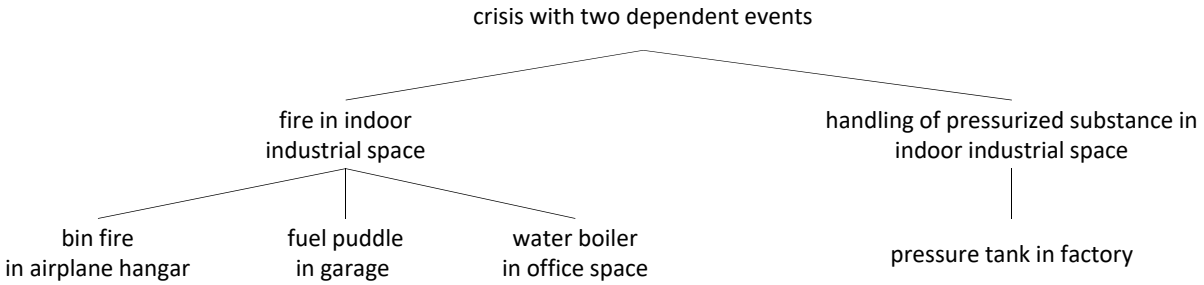

Figure 2. Levels of abstraction for vignettes

Example: handling of pressurized substance in indoor industrial space. The abstract training vignette "crisis with two dependent events" can be instantiated to "handling of pressurized substance in indoor industrial space" vignette, which can be further instantiated to a "handling of pressure tank in factory", where the initial event is the unintended build up of pressure in a tank, which has to be handled by opening a valve with a non-functional alternative of cutting the power to the tank, and where the secondary event is a tank rupture with ensuing dispersion of harmful material.

□

Fig. 2 shows the levels of abstractions in the examples. To summarize, think of a use case where an exercise manager at a National Disaster Training Center (NDTC) designs an abstract exercise template consisting of a "crisis with two dependent events". Then, handing this template over to a National Fire Training Center, a template "fire in indoor industrial space" is designed by the exercise manager there. The NDTC hands over its abstract template to a "Hazardous Materials Safety Agency". There, the NDTC's template is refined to a "handling of pressurized substance in indoor industrial space" template. Then, for different industries and organizations, a fire exercise manager consultant designs concrete vignettes (the lowest level in Figure 2. One for "bin fire in airplane hangar", one for "fuel puddle in garage", etc. Likewise, a pressurized material exercise manager consultant designs concrete vignettes for different organizations; here the "pressure tank in factory".

PLANNING ASSISTANT

The planning assistant (see Figure 1) gives automated support to exercise managers' planning activities.

The principal idea behind the planning assistant is that the problem of vignette composition can be seen as the combinatorial problem of finding configurations of objects satisfying certain causal constraints. The causal constraints describes interactions between the objects as well as the effects of actions; in other words the relationships between objects and the events that those relationship give rise to, as described in the examples above. The causal constraints then support the evolution of logically coherent plays exercising particular essential skills, such as situation awareness.

Now, if we also have a *computable* description of the causal constraints any vignette can be viewed as a dynamic system whose states are changed by events unfolding and trainee actions. The problem of generating vignettes then becomes the algorithmic problem of deriving possible lines of development based on the ability to predict the effects of different sequences of events and actions.

We need to distinguish between those plays in which events are handled successfully and not; success being determined by a fairly general criterion so as not to exclude plausible lines of development. For instance, preventing a fire from arising will count as a successful handling of an event, but so will putting the fire out, or pressing the alarm button when things really get out of hand. These plays will be scored differently according to the training objective and level of skill displayed by the trainee.

Purely unsuccessful developments consists of the plays in which the trainee's pursuits are futile; say the trainee searches for an extinguisher but never finds one, or pulls the plug of a pressure tank instead of opening the valve. All courses of action in this latter group will be scored at zero.

From an algorithmic point of view, therefore, the computation of coherent vignettes boils down to the generation of models that instantiate the causal effects of events and actions, and to deciding whether a sequence of such models belongs to the successful or unsuccessful group of plays.

Answer Set Programming (ASP) (Lifschitz 2008), is a declarative knowledge representation language that fits this bill nicely. It is a rule-based language that has its roots in deductive databases, logic programming, and automated

reasoning. ASP is oriented toward difficult (primarily NP-hard) search problems, which means that it follows a generate-and-test paradigm for finding solutions to computational problems: the algorithm selects or guesses a set of facts that may or may not hold in a satisfying model and uses the constraints expressed in the ASP program to prune the search space. Writing an answer set program involves identifying objects and simple facts, and codifying the relationship between them in the form of deductive rules. These rules are passed on to an *answer set solver* that generates models satisfying the constraints expressed by the rules. What we are proposing is that a simulation vignette can be seen as a coherently related sequence of such models or as an initial model that develops over time.

Using ASP for generating vignettes involves three things: first the vignette type must be represented as a *dynamic domain*, that is, as a temporal domain in which states evolve into other states *over time*. Secondly, one must have a *causal theory* that distinguishes valid state transitions (applying an extinguisher puts out the fire) from invalid ones (applying an extinguisher opens the door). Thirdly, one needs a *planner* that, given a dynamic domain, a causal theory and a success criterion, is able to derive all sequences of models representing successful courses of action. We describe each of these components in more detail below:

A dynamic domain is essentially a logical theory that is *temporalized*; a prerequisite for events-based simulations. This means, that all state-action-state transitions that can be deduced from the rules of an ASP program are indexed with respect to points in time. This is usually implemented by using a simple counter to keep track of time, and by adding an explicit temporal parameter to predicates and rules.

A causal theory consists of a general and a domain-specific part. The domain-specific part encodes the interaction between objects according to their relationships. It thus consists of causal rules such as "CO^2-based extinguisher *can put out fire in* oil-drenched paper". The general part of the causal theory consists of axioms characterizing the notion of causality as such. In other words, it is concerned with solving two well-known problems of symbolic AI: the first is the representation of *default rules*, also known as exception-allowing rules (Reiter 1980). Default rules are rules that capture the *typical* behaviour of objects and the typical effects of events, while allowing exceptions that render a particular rule inapplicable. An example of a default rule would be that oil-drenched paper typically burns when lit, with an exception that water-drenched such paper does not. Essentially, default rules allows one to reason about causality in a manner that adapts to changing circumstances. The second problem is the notorious *frame problem* (Shoham 1987). It concerns *inertia*, the general tendency for things to stay as they were unless acted upon. For instances, a socket, once unplugged stays unplugged. As inertia is a general characteristic of any physical domain, the causal theory includes general axioms that by default propagate the truth of facts through time.

The planner. In classical planning (Russell and Norvig 2010), a *goal* is a set of facts that characterize a stipulated optimal state of affairs. A *goal state* is any state that satisfies the goal. A *plan* is a sequence of events that steps through time in singleton increments taking the system from an initial state to a state satisfying the goal. A *solution* to a classical planning problem is a sequence of events that links the initial state to some goal state.

These areas are well researched and understood, and there are standard ways in ASP of representing and reasoning about them. In order to apply this to the automatic generation of vignettes all that needs to be done, in principle, is to identify coherent plays with solutions to planning problems. What we are proposing is essentially to treat vignette generation as a classical planning problem based on a dynamic domain expressed in terms of a causal description of actions and objects. This is the simple picture that emerges in the limiting case where there is only one type of vignette.

However, as explained in the previous section, we aim to support exercise planning at various levels of abstraction. Recapitulating briefly, a governmental exercise manager, in charge of overseeing national crisis training standards, may for instance wish to design exercises at the abstract level as templates to be refined by domain experts in various sectors and at various work places. Levels of abstraction are meant to facilitate efficient tool design and representation, and to enable exercise managers to plan a training session or an exercise at the level of object classes, as well as at the level of particular objects.

Referring back to Figure 2, the exercise management tool may offer a generic "crisis with two dependent events" vignette template, which may be refined to a "fire in industrial space" vignette template that, depending on user supplied constraints, may be refined into to various combinations of primary causes of fire (such as oil-drenched paper and water boilers), countermeasures (such as extinguishers, buttons and sprinklers), secondary fire hazards (air craft, fuel tank, ventilation unit) and so on.

The goal is to have the ASP engine support the instantiation procedure by coming up with vignettes, indicating possible developments of the initial situation that maximizes training value according to different objectives and metrics. Therefore, it is necessary to extend the ASP plan generation procedure described above to a more generic one capable of supporting automatic vignette generation based on templates.

To do this, we add an *ontology* (Gruber 1993) to the three aforementioned ASP components. An ontology is here understood as set of shared concepts for a domain that encodes their properties and the relations between them. Ontologies for the crisis response domain have been elaborated earlier; see e.g., (De Nicola et al. 2019; Steel et al. 2008; Bénaben et al. 2008). Our intent is not to develop a new ontology, but to demonstrate how ontologies provide the necessary shared machine-readable vocabulary for the ExManSim architecture.

The ontology will classify objects into classes (causes of fire, countermeasures, etc). The causal theory will then have to be modified so that it is expressed in terms of these classes, in addition to in terms of particular objects. Thus, the rules "initial fire *can ignite* oil-drenched paper" and "short circuit *can ignite* water boiler" there is a single generic rule saying "fire starter *can ignite* initial flammable material". Or even more abstractly for the uppermost level of Figure 2: "hazard *can trigger* initial crisis".

In the causal theory objects and relationships will constitute a *causal equivalence class* at various levels of abstraction, which, pertaining to the example at hand, means that, in successful plays, *intended means to handle the crisis* will all quench an *initial crisis* if applied at a sufficiently early (possibly variable) point in time.

STAGE AND CONTENT GENERATOR

The stage and content generator (see Figure 1) generates a stage and fills it with objects and relationships (content) at varying degrees of abstraction. It generates code structures according to ASP specifications at various levels of abstraction. The uppermost level of Figure 2 corresponds to an (almost) empty stage, while the lower levels add objects and relationships at successively more specific detail.

Recurrent training is necessary for learning, but for training decision making, environmental familiarity needs to be invariant to the training goals to avoid testing effects (Shadish et al. 2002). The stage and content generator must, with each training iteration, create a stage in which vignettes, objects and stage layout can be altered significantly, so that trainees cannot depend on their previous environmental knowledge.

The challenge is that the dynamically created stages and content must be generated rapidly, and optimally during training, to optimize motivation, recall and learning (Ericsson 2006a; Shadrick and Lussier 2009).

The stage and content generator will therefore support rapid generation of successive stages and content, at different levels of abstraction, by implementing optimization methods for automated generation of the next stage.

The stage and content generator will translate the causal theory derived by ASP into code structures using the ontology, so that the vignette objects will have the specified functionality. For this, code generation uses parent classes and inheritance, detailing each lower level of abstraction. Code generation will be recursive, where each reduction of abstraction translates more detail from the causal theory. The generated code is then translated into a format to be interpreted by a game engine to implement the simulation.

Generating the stage and content – the highest level of abstraction

At the highest level of abstraction of Figure 2, the abstract objects of a stage will not have any visual information. Instead, the information available will be formalized so that in lower levels of abstraction there can be more complex feature descriptions for object placement within the stage.

A *feature description* "assigns quantitative attributes to the detected features" (Gonzalez and Woods 2018). Feature descriptions are the rules by which the optimization algorithm abides when generating a stage and the metric by which the optimization assesses its performance. As an example, if a vignette object has a feature description that specifies close proximity to a wall, the closer a proposed stage solution places the object to a wall, the better the solution is considered. The optimization algorithm for stage creation will use all defined feature descriptions.

At this level of abstraction, there will be a list of feature descriptions describing the most basic of stage requirements. The generated code will be an abstract parent class which simply describes a class with two events and functional stubs for an *initial crisis*, a *secondary crisis* and *intended means* and *non-functional means* to handle the crises.

Generating the stage and content – the middle levels of abstraction

Referring to Figure 2's middle level of abstraction, "fire in indoor industrial space" and "handling of pressurized substance in indoor industrial space" have clearer definitions.

The detailing to "industrial space" allows the stage generator to create layout optimization solutions based on more detailed feature descriptors available for an industrial space according to the ontology. Relevant feature descriptors may include: indoor space, number of rooms, exit points, number of objects etc.

Now the first stage layout optimization attempts can be made. The stage generator would at this point use placeholder objects for representation. As an example, a stage solution of an industrial space may be generated that has two rooms. One of the feature descriptions defined for the placement of two objects describes a minimal distance between the two, and that they may not be in line of sight of each other. The optimized solution should propose placing the objects in separate rooms from. This would conform to the available feature descriptors. At this stage placeholders would be used for visual representation.

Creating a library of objects with feature descriptors for each possible object is laborious. Therefore, we aim to develop methods, where exercise managers can create custom stage layout templates using a level editor with drag-and-drop mechanics. When the desired template is created, the user can select objects, walls and rooms and choose relationships as feature descriptions. An exercise manager could drag an object into their custom template stage near a wall and select as a feature description, 'distance to the nearest wall'. This object receive this feature description. When a stage gets generated, it will attempt to place this object near a wall to adhere by the feature description. The idea is that an exercise manager can create a completed template that conforms to their expectations of an area in real life and then select specifically what feature descriptors the objects in the stage should have. When the stage is generated it should have a degree of variation in its proposed solution each time, while adhering to a set of defined rules that allow for a realistic layout. This also allows the system to use an exercises manager's domain specific knowledge in stage generation.

The code generation to translate the now more detailed ontology and causal theory to classes and functionality can be expanded. The code generator can create a new child class which inherits from the previously defined crisis parent class , into "fire in indoor space" and "pressurized substance in indoor space". The ontology now describes two objects, which can be created from an abstract parent class which holds basic object information. The causal theory at this stage has defined the functions of both objects: being on fire and pressure accumulation with potential explosion. Variables and functions that cover this functionality can be placed within newly defined object child classes. The new crisis child class can create and hold objects from the two newly defined child object classes. This is illustrated simplistically in Figure 3.

Generating the stage and content – the lowest level of abstraction

The lowest tier of Figure 2 shows concrete stage descriptions: "fuel puddle in garage" and "pressure tank in factory". The stage description is now clearly defined: an airplane hangar, office, garage, etc. It may be that the previously defined stage layout solutions fit perfectly with the new layout feature descriptors. If it does not, it is necessary for the the stage and content generator to start proposing new layout solutions more. As an example, if a previous proposed solution of the industrial space at a higher level of abstraction consisted of one large room, this may conform to the features descriptions required of an airplane hangar, garage or factory. The previous solution can be

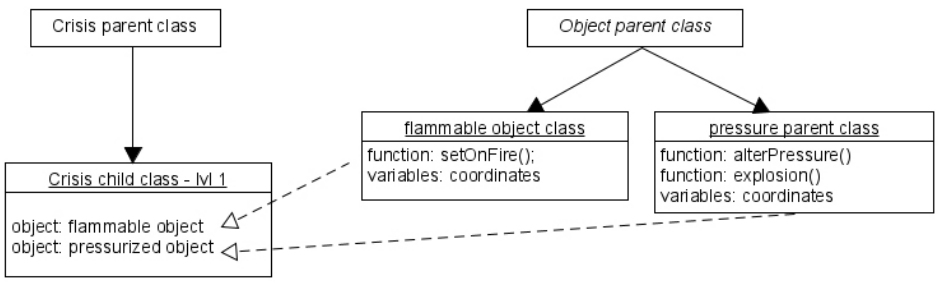

Figure 3. Code generation in mid level of abstraction

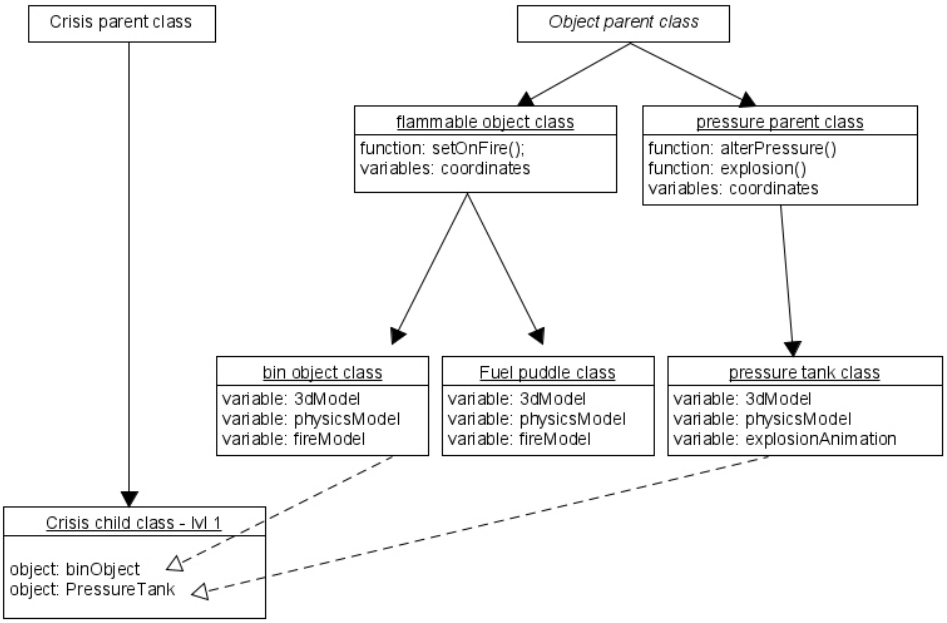

Figure 4. Code generation in lowest level of abstraction

immediately utilized. If the new requirement is that of an office space, however, the proposed room layout may not conform to the desired feature descriptions, as an office space may require many small rooms instead of one large one. The stage generation software would calculate and propose a new layout solution.

Similar adaptations would occur with objects: In the industrial space solutions, the placement of a large industrial object near the center of the large room, appropriate for a hangar, garage or factory, may have to be altered for an office space. The objects that do not adhere to the desired feature descriptions for type, size and relative placement, would have to be moved or replaced and new layout solutions would have to be generated.

The vignettes are also more clearly defined. Previously vignette placeholder objects were assigned possible locations in the training stage. If the required feature descriptions for these specified vignettes no longer conform to their placements in the previously proposed solutions, the stage and content generator will be required to calculate new placement solutions within the stage.

The code generation for the ontology and the causal theory can then proceed to its most detailed form. The flammable and pressurized object are now determined (bin, fuel puddle, water boiler, pressure tank). A final child class can be created where functionality can be expanded to fit the latest causal theory, and models can be chosen which are used to represent the objects in the simulation. The objects used in the crises child class can then be replaced with the more detailed child object classes. This new class structure that the code generator will make is illustrated simplistically in Figure 4.

With the feature descriptors now defined at the lowest level of abstraction, the stage and content generator can aim to create a finalized solution. This supplies trainees with a stage that includes a realistic training space, placement of objects and vignettes whose features and behavior are defined by the generated code. The generated code is automatically put into a script for use within the game engine which creates the simulation.

HYPOTHESES FOR FUTURE WORK

We are now ready to state the main research hypothesis for the next stage of this work, which pertains to the planning-to-simulation direction of the ExManSim architecture (left-to-right arrow in Figure 1):

> *The steps from planning to simulation through the planning assistant and the stage and content generator can be automated by reasoning in various ways over a computable representation of the salient causal relations between objects in the training domain.*

More specifically, the planning assistant and the stage and content generator correspond to two different reasoning tasks supported by an answer set representation of a causal theory of the training domain. This idea is illustrated in Figure 5, which is adapted from (Shanahan 1999).

The two reasoning tasks can be broadly categorised into *deductive tasks* and *abductive tasks*. In a deductive task, "what happens when" and "what actions do" are given, and the task is then to determine "what's true when". Deductive tasks include *temporal projection or prediction* (Shanahan 1999, p. 2). In a nutshell, this mode of reasoning takes a set of rules that express the causal effects of actions together with a specification of which actions are performed when and derives a complete description of the state of the domain after those actions are performed.

In an *abductive task*, in contrast, "what actions do" and "what's true when" are supplied, and the task is to determine "what happens when". In other words, a sequence of actions is sought that leads to a given outcome. Abductive reasoning includes explanation, diagnosis, and planning (Shanahan 1999). It is a matter of taking the causal rules together with a specification of what is true when in order to come up with a sequence of actions that explain the evolution of those facts.

Both of these problems are well-studied, also in the context of ASP; see e.g. (Gelfond and Kahl 2014). We add the observation that the proposed planning assistant can be understood as an abductive reasoning component whereas the stage and content generator can be understood as a deductive one. The planning assistant suggests and validates events and vignettes to ensure that the exercise manager designs events in a logically consistent manner in line with the essential skills to be trained. Similarly, the stage and content generator translates answer set programs to code structures to be interpreted by simulation engines, so that the exercise manager's design changes are reflected instantly in the simulations; which is a deduction problem.

This framework is theoretically simple and appealing but needs to be generalized if it is to cater for a large enough variety of training domains and avoid the familiarity effects associated with repetitive training. We proposed that a sufficient measure of variety and freedom of choice can be implemented by way of an ontology. The ontology lifts the causal theory from objects to classes so that the causal rules are formulated with respect to, not primarily particular objects, but rather sets of causally equivalent ones.

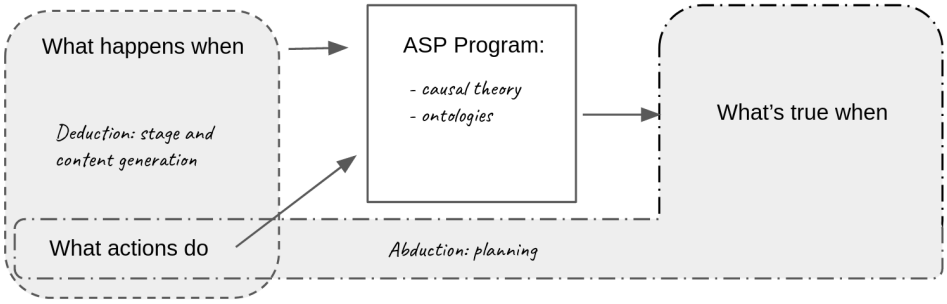

Figure 5. Illustration of the research hypothesis

FINAL REMARKS

Answering calls for tool support for deliberate practice in simulation-based training, we argue that a service concept based on skill-targeted events, rather than on objects and their interactions, will allow for the rapid composition of training vignettes.

We are using Answer Set Programming to create logical structures for rapid vignette generation, at various levels of abstraction. Thus, abstract vignettes can be used as the bases to generate successively more concrete and detailed vignettes, reflecting exercise managers' various levels of planning. Corresponding code will be generated that translates the Answer Set Programming specifications to classes and functions to be interpreted by simulation engines. This allows exercise planning and re-planning and also gives tool support for varying training vignettes to counteract familiarity effects during recurrent training. Automated generation will rely on optimisation algorithms that use quantitative descriptions of objects and their relation to the stage.

Our aim is to develop an exercise management system that addresses current shortcomings in simulation-based training, in line with what we have described in this article. This system is under incremental development, and small pieces of integral functionality will be released to a reference group consisting of crisis response professionals for concept validation, refinement and adjustment.

Acknowledgements

This research is funded by the Research Council of Norway under project no. 282081 "MixStrEx". The authors are grateful to the anonymous reviewers.

REFERENCES

Armstrong, J. S., ed. (2001). *Principles of Forecasting: A Handbook for Researchers and Practitioners*. Kluwer Academic Publishers.

Asprusten, M. and Hannay, J. E. (2018). "Simulation-Supported Wargaming using M&S as a Service (MSaaS)". In: *Proc. NATO Modelling and Simulation Group Symp. on Multi-National Pooling and Sharing of Simulation Resources under the M&S as a Service Paradigm (STO-MP-MSG-159)*.

Bénaben, F., Hanachi, C., Lauras, M., Couget, P., and Chapurlat, V. (2008). "A metamodel and its ontology to guide crisis characterization and its collaborative management". In: *Proc. 5th Int'l Conf. Information Systems for Crisis Response and Management (ISCRAM)*, pp. 189–196.

De Nicola, A., Melchiori, M., and Villani, M. L. (2019). "Creative design of emergency management scenarios driven by semantics: An application to smart cities". In: *Information Systems* 81, pp. 21–48.

Durlach, P. J. (2018). "Can we talk? Semantic Interoperability and the Synthetic Training Environment". In: *Proc. Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2006*. National Training and Simulation Association.

Edgren, M. G. (2012). "Cloud-Enabled Modular Services: A Framework for Cost-Effective Collaboration". In: *Proc. NATO Modelling and Simulation Group Symp. on Transforming Defence through Modelling and Simulation—Opportunities and Challenges (STO-MP-MSG-094)*.

Endsley, M. R. (2000). "Theoretical Underpinnings of Situation Awareness: A Critical Review". In: *Situation awareness analysis and measurement*. Ed. by M. R. Endsley and D. J. Garland. Lawrence Erlbaum Associates Publishers, pp. 13–32.

Ericsson, K. A. (2006a). "An Introduction to Cambridge Handbook of Expertise and Expert Performance: Its Development, Organization, and Content". In: *The Cambridge Handbook of Expertise and Expert Performance*. Ed. by K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman. Cambridge Univ. Press. Chap. 1, pp. 3–20.

Ericsson, K. A. (2006b). "The Influence of Experience and Deliberate Practice on the Development of Superior Expert Performance". In: *The Cambridge Handbook of Expertise and Expert Performance*. Ed. by K. A. Ericsson, N. Charness, P. J. Feltovich, and R. R. Hoffman. Cambridge Univ. Press. Chap. 38, pp. 683–703.

Gelfond, M. and Kahl, Y. (2014). *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press.

Gigerenzer, G. and Todd, P. M., eds. (1999). *Simple Heuristics that Make Us Smart*. Oxford University Press. Chap. 4, pp. 75–95.

Gonzalez and Woods (Jan. 2018). *Digital Image Processing*. Fourth Edition. Pearson, pp. 811–812.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications". In: *Knowledge Acquisition* 5.2, pp. 199–220.

Grunnan, T. and Fridheim, H. (2017). "Planning and conducting crisis management exercises for decision-making: the do's and don'ts". In: *EURO Journal on Decision Processes* 5, pp. 79–95.

Hannay, J. E. and Kikke, Y. (2019). "Structured crisis training with mixed reality simulations". In: *Proc. 16th Int'l Conf. Information Systems for Crisis Response and Management (ISCRAM)*, pp. 1310–1319.

Hannay, J. E. and van den Berg, T. W. (2017). "The NATO MSG-136 Reference Architecture for M&S as a Service". In: *Proc. NATO Modelling and Simulation Group Symp. on M&S Technologies and Standards for Enabling Alliance Interoperability and Pervasive M&S Applications (STO-MP-MSG-149)*.

Kahneman, D. and Frederick, S. (2004). "A Model of Heuristic Judgment". In: *The Cambridge Handbook of Thinking and Reasoning*. Ed. by K. J. Holyoak and R. G. Morrison. Cambridge Univ. Press, pp. 267–294.

Lifschitz, V. (2008). "What is Answer Set Programming?" In: *Proc. 23rd National Conference on Artificial Intelligence (AAAI'08) – Volume 3*. AAAI Press, pp. 1594–1597.

Pollestad, B. and Steinnes, T. (2012). "Øvelse gjør mester?" MA thesis. University of Stavanger, Dept. of Media and Social Sciences.

Reiter, R. (1980). "A logic for default reasoning". In: *Artificial Intelligence* 13.1, pp. 81–132.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Third. Series in Artificial Intelligence. Prentice Hall.

Salas, E., Wildman, J. L., and Piccolo, R. F. (2009). "Using Simulation-Based Training to Enhance Management Education". In: *Academy of Management Learning & Education*.

Shadish, W. R., Cook, T. D., and Campbell, D. T. (2002). *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin.

Shadrick, S. B. and Lussier, J. W. (2009). "Training Complex Cognitive Skills: A Theme-based Approach to the Development of Battlefield Skills". In: *Development of Professional Expertise*. Ed. by K. A. Ericsson. Cambridge University Press. Chap. 13, pp. 286–311.

Shanahan, M. (1999). "The Event Calculus Explained". In: *Artificial Intelligence Today: Recent Trends and Developments*. Ed. by M. J. Wooldridge and M. Veloso. Springer, pp. 409–430.

Shoham, Y. (1987). "What is the frame problem?" In: *The Frame Problem in Artificial Intelligence*. Ed. by F. M. Brown. Morgan Kaufmann, pp. 5–21.

Simulation Interoperability Standards Organization (2018). *SISO-GUIDE-006-2018 – Guideline on Scenario Development for Simulation Environments*.

Skarpaas, I. and Kristiansen, S. T. (2010). *Simulatortrening for ny praksis: Hvordan simulatortrening kan brukes til å utvikle Hærens operative evne*. Tech. rep. Work Research Institute.

Steel, J., Iannella, R., and Lam, H.-P. (2008). "Using ontologies for decision support in resource messaging". In: *Proc. 5th Int'l Conf. Information Systems for Crisis Response and Management (ISCRAM)*, pp. 189–196.

Steen-Tveit, K. and Jaziar, R. (2019). "Analysis of Common Operational Picture and Situational Awareness during Multiple Emergency Response Scenarios". In: *Proc. 16th Int'l Conf. Information Systems for Crisis Response and Management (ISCRAM)*.

Tolk, A. (2012). "Integration of M&S Solutions into the Operational Environment". In: *Engineering Principles of Combat Modeling and Distributed Simulation*. Ed. by A. Tolk. Wiley. Chap. 15, pp. 295–327.

van den Berg, T. W., Huiskamp, W., Siegfried, R., Lloyd, J., Grom, A., and Phillips, R. (2017). "Modelling and Simulation as a Service: Rapid deployment of interoperable and credible simulation environments – an overview of NATO MSG-136". In: *Proc. 2017 Fall Simulation Innovation Workshop*. 17F-SIW-018.