# Using Block-Based Programming and Sunburst Branching to Plan and Generate Crisis Training Simulations

Dashley K. Rouwendal van Schijndel[1](✉), Audun Stolpe[2], and Jo E. Hannay[2]

[1] Department of Technology Systems, University of Oslo, Pb. 70, 2027 Kjeller, Norway
d.k.rouwendal@its.uio.no

[2] Department of Applied Research in IT, Norwegian Computing Center, Pb. 114 Blindern, 0314 Oslo, Norway
{audun.stolpe,jo.hannay}@nr.no

**Abstract.** Simulation-based exercises for crisis response are difficult to plan. We suggest an intuitive planning interface where exercise managers can use block-based programming to create machine-readable training vignettes. An answer set programming module interprets these vignettes and generates all possible actions and causal relations, which are then visualised for the exercise manager in a sunburst diagram. This allows exercise managers to create training vignettes using visual techniques and to see the possible results and causal relations of the trainees' actions. This facilitates using exercise results to adjust further vignette design.

**Keywords:** Simulation-based training · Block-based programming · Answer set programming · Visual planning

## 1 Introduction

Simulation-based crisis response exercises and training within military and civilian organisations are often complex, and it has proven difficult to design exercises with clear learning objectives and targeted building and assessment of skills; see for example, [4,11,14]. To improve on this situation, an *Exercise Management & Simulation* architecture (ExManSim) was proposed, where the organisational and technical challenges concerning structured planning, execution and analysis of exercises are explicitly addressed [5].

Existing systems for simulation-based training often focus on what objects and interactions should be present in simulations, rather than on what skills should be stimulated and measured [5,6]. In contrast, a central point in the ExManSim architecture is to offer skill-stimulating events as basic building blocks, so that exercise managers can compose simulations in terms of learning effects, rather than merely in terms of objects and their interactions.

This demands tool support that abstracts away details, so that the exercise manager (trainer) can compose training vignettes (a series of events) designed to

© Springer Nature Switzerland AG 2020
C. Stephanidis and M. Antona (Eds.): HCII 2020, CCIS 1226, pp. 463–471, 2020.
https://doi.org/10.1007/978-3-030-50732-9_60

stimulate the desired skills. ExManSim focuses on decision-making skills, such as situational awareness, resource coordination and collaboration; e.g., [2]. We propose to use block-based programming; a visual programming methodology for composing computer programs from blocks that represent predefined functionality. Here, we will use blocks designed to compose events into vignettes.

After the exercise manager has composed a vignette using block-based programming, the vignette object information is exported to an answer set programming (ASP) module. ASP allows for the creation of knowledge representation models that can be stored in machine-readable formats. In our system, ASP runs behind the scenes and uses the object information and basic predefined vignette parameters to calculate valid possible sequences of events and their possible outcomes. This stored information is then used to implement a simulation semi-automatically on a virtual simulation platform.

In time, data collected from the trainees' actions and interactions within the vignettes will be used with ASP to create new block-based designs for follow-up vignettes that focus on training aspects that need special attention. In this manner, the system supports deliberate practice [3] and adaptive thinking [12,13] in order to maximise learning.

## 2 Planning at Several Levels of Abstraction

Based on workshops with crisis response practitioners and researchers, we are designing ExManSim to facilitate planning at several levels of abstraction. Referring to Fig. 1, think of a use case where an exercise manager at a National Disaster Training Centre (NDTC) designs an abstract exercise template consisting of a "crisis with two dependent events". Then, handing this template over to a National Fire Training Centre, a template "fire in indoor industrial space" is designed by the exercise manager there. The NDTC also hands over its abstract template to a "Hazardous Materials Safety Agency". There, the NDTC's template is refined to a "handling of pressurised substance in indoor industrial space" template. Then, for different industries and organisations, a fire exercise manager consultant designs concrete vignettes (the lowest level in Fig. 1. Here, one for "bin fire in airplane hangar", "fuel puddle in garage", and "stove fire in industrial kitchen", etc. Likewise, a pressurised material exercise manager consultant designs concrete vignettes; for example "pressure tank in factory".

## 3 Vignette Design Using Block-Based Programming

Human beings are more efficient at processing visual information than textual representation; for pointers, see [10]. Block-based programming is a popular visual method to compose computer programs. People who have no prior skills in computer programming can easily compose programs from predefined pieces (blocks) of functionality. These blocks are presented visually in menus and can be specialised for various uses in actual programs. Blocks are then assembled in

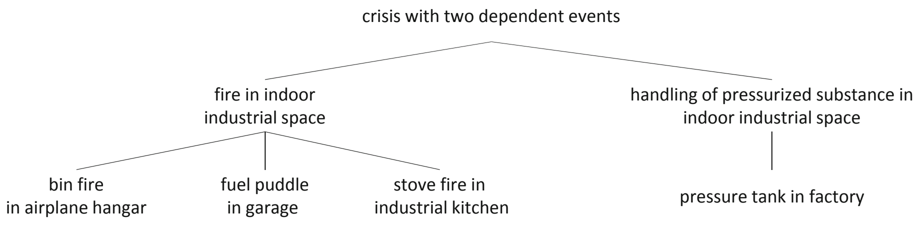
crisis with two dependent events
fire in indoor industrial space
handling of pressurized substance in indoor industrial space
bin fire in airplane hangar
fuel puddle in garage
stove fire in industrial kitchen
pressure tank in factory

**Fig. 1.** Levels of abstraction for vignettes

a graphical workspace to form a computer program. This technique is now used in various products; such as Micro:bit[1] and Tynker[2].

Block-based programming reduces the rate of syntax-errors and increases efficiency for individuals who are non-frequent programmers [7]. It makes programming easier, since the coding options are available visually as in show in Fig. 2. It is easier to create a system via recognition (since the blocks appear in a selection menu) rather then by recall [9].

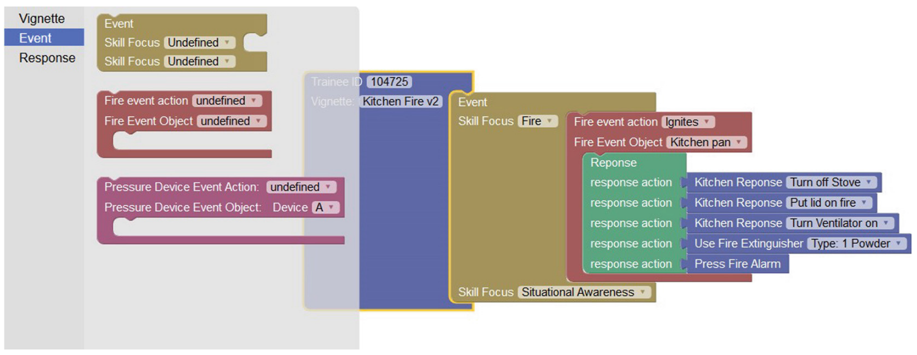
Vignette
Event
Response
Event
Skill Focus Undefined
Skill Focus Undefined
Fire event action undefined
Fire Event Object undefined
Pressure Device Event Action: undefined
Pressure Device Event Object: Device A
Trainee ID 104725
Vignette: Kitchen Fire v2
Event
Skill Focus Fire
Fire event action Ignites
Fire Event Object Kitchen pan
Reponse
response action Kitchen Reponse Turn off Stove
response action Kitchen Reponse Put lid on fire
response action Kitchen Reponse Turn Ventilator on
response action Use Fire Extinguisher Type: 1 Powder
response action Press Fire Alarm
Skill Focus Situational Awareness

**Fig. 2.** Recognition-based vignette construction

Block-based programming is designed to visualise abstract syntax constructions. We hold that a block-based visual programming language will also provide crisis management personnel, who may have little or no simulation-technical skills, with an intuitive tool for designing training vignettes. We use the generic Blockly code library[3] that allows one to create customised block languages.

In the example in Fig. 3, the exercise manager has been tasked to design training for a stove fire in industrial kitchen (bottom level of Fig. 1) and has been handed a template for fire in indoor industrial space (middle level of Fig. 1) by

[1] https://microbit.org/.
[2] https://www.tynker.com/.
[3] https://developers.google.com/blockly.

the National Fire Training Centre. He is therefore presented with the template blocks for fire in indoor industrial space training pre-loaded in the vignette design work space; see leftmost part of Fig. 3. At this level of abstraction, the type of fire and the skill to be trained has not been selected, and the possible response actions have not been selected. The exercise manager can then proceed to detail the template blocks with blocks and variables for the particular fire response training; in this case, a stove fire. In the rightmost part of Fig. 3, the exercise manager selects blocks for a pan fire which ignites as the first crisis event. The skill to be trained is situational awareness. Concrete responses are added as blocks to the response template block and includes the option for the trainee to turn off the stove, put a lid on fire (pan), turn the ventilator on and/or use a powder-based fire extinguisher.

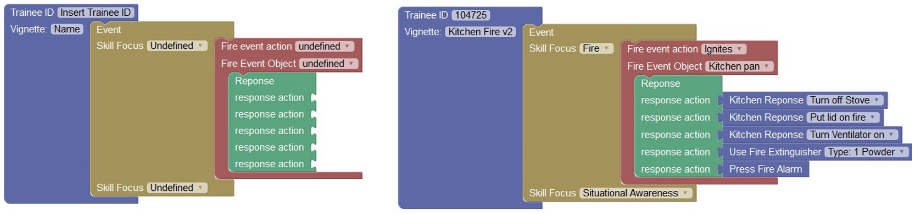
Trainee ID Insert Trainee ID
Vignette: Name
Event
Skill Focus Undefined
Fire event action undefined
Fire Event Object undefined
Reponse
response action
response action
response action
response action
response action
Skill Focus Undefined
Trainee ID 104725
Vignette: Kitchen Fire v2
Event
Skill Focus Fire
Fire event action Ignites
Fire Event Object Kitchen pan
Reponse
response action
response action
response action
response action
response action
Kitchen Reponse Turn off Stove
Kitchen Reponse Put lid on fire
Kitchen Reponse Turn Ventilator on
Use Fire Extinguisher Type: 1 Powder
Press Fire Alarm
Skill Focus Situational Awareness

**Fig. 3.** Abstract fire vignette (left), defined fire vignette (right)

Block-based programming serves a dual purpose. First, a custom block representation of a vignette acts as a grammar. It presents the user with a basic vocabulary in the form of elementary blocks. These blocks are typed, and constraints on the types determine the well-formedness of the vignette. Moreover, the workspace has information devices in the form of pull-down and pop-up menus that allows one to specialise and configure blocks for specific purposes when working at lower levels of abstraction (Fig. 1). Together, the type system, the constraints and the information devices represent an approach to syntax learning based on direct recognition rather than recall.

Secondly, block-based programming offers a feature to convert composite blocks into custom code or text. We exploit this by wiring each block to a piece of ASP code. The vignette grammar ensures that these code snippets are compositional and add up to a logical description in ASP of the initial situation of the training exercise. In turn, this description is fed as input to a causal theory, also encoded in ASP, that encodes the interaction between the objects as well as the effects of actions in the training domain. The reason for this is described in more detail in the next section.

## 4 ASP and Model Output Visualisation

The block design space is hooked into an ASP representation of the salient causal relations in the training domain. Then, ASP will compute the coherent plays, if any, that the selected set of blocks and properties support. In fact, the computation of the vignette provides two different reasoning services. One is *validation*: not all block puzzles yield playable vignettes even if they are syntactically correct. For instance, the vignette designer may decide to include a wet chemical extinguisher if the vignette involves a fire in kitchen grease, since wet chemical extinguishers are designed for that. He may also decide to include a foam extinguisher, which is not appropriate for grease fires, to allow the trainee to make mistakes. However, if the designer does not select at least one effective extinguisher, then the vignette is not successfully playable. Such a vignette with an unattainable goal is *invalid* and will be rejected by the ASP computation. The second reasoning service is to *deduce* or *generate* the different ways in which the initial situation can evolve according to the causal theory. If presented in a suitably friendly format, the vignette designer can gain a sense of the depth and training value of his current block layout and can revise it when necessary.

The details of the answer set programming module itself is not the focus of this paper. Rather, the focus is the visual user interface of the vignette design space. The challenge from a user interface point of view is that the number of ways a situation can develop from even a simple causal theory is usually too large to be conveniently represented by a traditional rooted top-down or bottom-up tree widget. More often than not, certainly for most interesting vignettes, this is going to be a very bushy tree with a high branching factor. Our experiments with decision trees and other tree-like structures indicate that the user will have to scroll through very large visualisations to get an overview of the possible outcomes; in particular to view the leaf nodes, where the success or failure of a course of action is recorded.

In order to be useful to the designer, the plays or models generated by ASP should be visualised in a manner that more effectively conveys the training potential of the vignette. As a minimal requirement, the visualisation should display all the choice points; that is, all the different courses of action a trainee may opt for and all the outcomes of the different plays; distinguishing successful from unsuccessful ones in a single window frame.

An evolution diagram frequently used in biology for representing phylogenetic trees [8] is the sunburst diagram.[4] A phylogenetic tree represents evolutionary relationships among organisms. The pattern of branching in a phylogenetic tree reflects how species or other groups evolved from a series of common ancestors, which means that the branching factor is usually considerable. A sunburst diagram (also known as a ring chart, multi-level pie chart, belt chart and radial treemap) shows hierarchies through a series of rings, that are sliced for each category node. Each ring corresponds to a level in the hierarchy, with the central circle representing the root node and the hierarchy moving outwards from it. The growth in size of the number of rings is constant, irrespective of the branching factor, since a single ring encodes all parent-child relationships in that level of the tree. Also, due to its concentric 360° nature, the sunburst diagram maximizes the use of space within a single frame.

[4] https://datavizcatalogue.com/methods/sunburst_diagram.html.

Figure 4 shows a trimmed down excerpt of some ways in which the kitchen fire vignette evolves, according to the ASP encoding of a causal theory relating, stoves, pans, grease, fans, extinguishers, and so on.

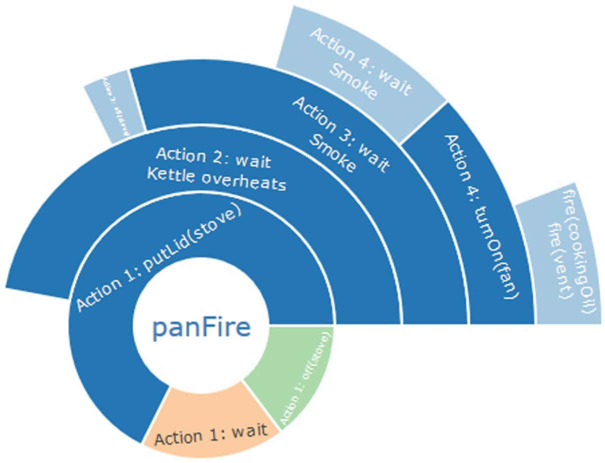

**Fig. 4.** Reduced example of a sunburst diagram for the kitchen fire vignette.

The particular sunburst diagram in Fig. 4 is from the Plotly open source graphing libraries.[5] It has several attractive features that makes it a good fit for our purposes. Not only is it capable of representing, for most realistic cases, sufficiently large number of choice points, branches and outcomes in a single window pane, it also adjusts the centre of the diagram interactively. That is, when the user clicks on a segment other than the centre, the sunburst chart can rearrange itself to make that segment the new centre. Thus, irrespective of the size of the data set, scrolling is never needed for exploration.

Finally, note that the sunburst representation of vignettes can be made to support both of the reasoning services mentioned above; that is both exploration and validation. Using a colouring scheme that labels permitted states and actions in green and prohibited ones in red (cf. [1]), an invalid vignette is one with an entirely red outer rim. This colouring scheme offers a way of designating a vignette as invalid, while still developing available of information about it. The point at which a story or play becomes red, for instance, is valuable information for debugging the Blockly layout of the vignette. This is ongoing research.

## 5 The Feedback Loop

A further purpose of the ExManSim architecture is that after a training has been completed by a trainee, the system can reset and the vignette design can be

[5] https://plot.ly/graphing-libraries/.

adjusted according to the trainees' performance. The trainees can then undergo the training once more with the vignette adapted to their new skill levels.

The data collection that takes place during training will give valuable insight of what the future focus of the trainees' training should be. In an example of the kitchen fire, it may be that the trainees are not sufficiently knowledgeable about different types of fire extinguishers, and therefore in the next training round there could be more focus on this particular facet of training. The premise is that you can automate this training adaptive system via the ASP module.

A topic that we are actively researching at the time of writing, is to use the green/red colouring scheme that was mentioned toward the end of the previous section for scoring plays. The green/red feature, a version of which we have already implemented, is a kind of *deontic overlay* that is grafted onto the causal theory. It describes sufficient conditions for classifying actions as either green (acceptable, legal) or red. The ASP engine propagates these conditions over complex states and transitions, based on the rules in the causal theory. In itself, this is not a new idea, but derives from (cf. [1]). What *is* novel, to the best of our knowledge, is the idea of using the deontic overlay for scoring plays. In the simplest case, this is matter of subtracting the sum total of red states and actions from the green ones to obtain a crude measure of the overall quality of the course of action pursued in that play.

The scores will be used for solution optimisation and dynamic assignment of difficulty levels, and will be applied in repeated rounds of training. This will give an effective record of the progress a trainee is making and give insight to the effectiveness of the various parts of the training overall. This addresses a basic challenge in simulation-based training.

## 6 Final Remarks

To summarise the process that ExManSim supports: Exercise regulatory bodies provide basic requirements and exercise templates. Domain-specific exercise managers are directed to a block-based vignette design tool where abstract block-based vignettes that represent the above exercise templates have already been created. The exercise manager specifies domain-specific block-based details as desired in the block-based template vignette. When this is completed, the block-based vignette design is passed in machine-readable format to an answer set programming module, which creates an outline of all the possible actions a trainee can take in the vignette. Scoring methods are built into the system to help assess a trainee's performance. The exercise manager will then get a visual overview of the answer set model in a sunburst diagram, where he can decide whether to accept the generated model or go back and adjust the vignette design. When the vignette is accepted, the virtual training scene and vignette is generated in a simulation, and the virtual training can take place. The data from the training is collected, and after the training has ended, the design and training cycle can repeat, but now with specific recommendations which can be applied to the vignette design for greater customisation toward the needs of the trainee.

We postulate that this feedback loop of training, performance analysis and retraining at the exercise manager's control through an easy-to-use, block-based vignette design methodology and real-time visualisation of the trainees possible actions, will allow for a form of simulation-based training that creates greater training efficiency, feedback and record keeping.

## References

1. Craven, R., Sergot, M.: Agent strands in the action language nC+. J. Appl. Logic **6**(2), 172–191 (2008). Selected papers from the 8th International Workshop on Deontic Logic in Computer Science
2. Endsley, M.R.: Theoretical underpinnings of situation awareness: a critical review. In: Endsley, M.R., Garland, D.J. (eds.) Situation Awareness Analysis and Measurement, pp. 13–32. Lawrence Erlbaum Associates Publishers, Mahwah (2000)
3. Ericsson, K.A.: The influence of experience and deliberate practice on the development of superior expert performance. In: Ericsson, K.A., Charness, N., Feltovich, P.J., Hoffman, R.R. (eds.) The Cambridge Handbook of Expertise and Expert Performance, pp. 683–703. Cambridge University Press, Cambridge (2006)
4. Grunnan, T., Fridheim, H.: Planning and conducting crisis management exercises for decision-making: the do's and don'ts. Eur. J. Decis. Process. **5**, 79–95 (2017)
5. Hannay, J.E., Kikke, Y.: Structured crisis training with mixed reality simulations. In: Proceedings of 16th International Conference Information Systems for Crisis Response and Management (ISCRAM), pp. 1310–1319 (2019)
6. Hannay, J.E., van den Berg, T.W.: The NATO MSG-136 reference architecture for M&S as a service. In: Proceedings of NATO Modelling and Simulation Group Symposium on M&S Technologies and Standards for Enabling Alliance Interoperability and Pervasive M&S Applications (STO-MP-MSG-149). NATO Science and Technology Organization (2017). paper 13
7. Holwerda, R., Hermans, F.: A usability analysis of blocks-based programming editors using cognitive dimensions. In: Proceedings of 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). IEEE Computer Society (2018)
8. Letunic, I., Bork, P.: Interactive tree of life (iTOL): an online tool for phylogenetic tree display and annotation. Bioinformatics Adv. Access **23**(1), 127–128 (2006)
9. Monig, J., Ohshima, Y., Maloney, J.: Blocks at your fingertips: blurring the line between blocks and text in GP. In: Proceedings of 2015 IEEE Blocks and Beyond Workshop. IEEE (2015)
10. Moody, D.L.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**(6), 756–779 (2009)
11. Pollestad, B., Steinnes, T.: Øvelse gjør mester? Master's thesis, University of Stavanger, Department of Media and Social Sciences (2012). in Norwegian
12. Pulakos, E.D., Arad, S., Donovan, M.A., Plamondon, K.E.: Adaptibility in the work place: development of a taxonomy of adaptive performance. J. Appl. Psychol. **85**(4), 612–624 (2000)

13. Shadrick, S.B., Lussier, J.W.: Training complex cognitive skills: a theme-based approach to the development of battlefield skills. In: Ericsson, K.A. (ed.) Development of Professional Expertise, pp. 286–311. Cambridge University Press, Cambridge (2009)
14. Skarpaas, I., Kristiansen, S.T.: Simulatortrening for ny praksis: Hvordan simulatortrening kan brukes til å utvikle hærens operative evne. Tech. rep. Work Research Institute (2010). in Norwegian