A 3D Security Modelling Platform for Social IoT Environments

Bo Zhou, Curtis Maines, Stephen Tang, Qi Shi, Po Yang, Qiang Yang, and Jun Qi

***Abstract*—Social IoT environment comprises not just smart devices, but also the humans to interact with these IoT devices. The benefits of such system are overshadowed by the issues of cyber security. A new approach is required for us to understand the security implication under such dynamic environment, while taking both the social and technical aspects into consideration. This paper proposed a 3D security modelling platform that can capture and model security requirements in Social IoT environment. The modelling process is graphical notation based, working as a security extension to Business Process Model and Notation. Still, it utilises the latest 3D game technology thus the security extensions are generated through the third dimension. In this way, the introduction of security extensions will not increase the complexity of the original SIoT scenario, while keeping all the key information in the same platform. Together with the security ontology we have proposed, these comprehensive security notations created a unique platform that aiming at addressing the ever complicated security issues in SIoT envorinment.**

***Index Terms*—Business process, game technology, notation, security modelling, social IoT.**

I. Introduction

With recent advances in technology, Internet of Things (IoT) will play an important role in our daily lives. Billions of small devices with the capability of computing and communications not just enable a new way of interactions between humans and smart things, but also change our behaviors in the society, for example through the social networks. These ubiquitous IoT devices deeply embedded into our lives and eventually form part of a Social IoT (SIoT) environment [1].

Like any existing IT systems, a paramount question needs to be answered by the SIoT is the security issue [2]. The ubiquitous nature of IoT means it is not just the digital contents on these devices will be facing the threat. Even humans' physical safety could be in danger due to the blurred boundary between the digital and physical worlds. The changing social relationship between humans and things represent the main difference, with the security implication is getting even more complicated. In such environment, technology solution alone will not be able to address all the security concerns. The social aspect of the issue must be taken into account as well [3]. Therefore, the socio-technical challenges in terms of security should be addressed properly before the real implementation of SIoT.

Among all the security issues, the very first one is always to identify and model the security requirements. In a dynamic environment like SIoT, devices may join and leave the network without the notice of the user. The lack of central point brought the question of where the security policy should be defined and how can they be enforced. In this paper, we investigate a novel approach to create a security requirement modelling platform that could help to effectively capture and model the security requirements in the SIoT environment. We build the platform based on Business Process Model and Notation (BPMN), which is comprehensive enough to represent both the humans and things as services in the same platform, and extend it with security requirements being captured in the third dimension. In this way, the security requirements are managed separately without increasing the complexity of the original SIoT scenarios.

With a pre-defined security ontology [4], we created a set of security notations that can serve as an extension to the existing BPMN library. Together, they can model a SIoT scenario and the security requirements in a two steps approach. The notations we created have taken the Moody's "Physics of Notations" [5] into account and they are the first security notations to satisfy most of the design principles.

The rest of paper is organised as the follows. Section II introduces the background of using BPMN to model SIoT scenario and the key concepts of the security ontology we use. The current modelling approaches and benefit of using third dimension are discussed in Section III. In Section IV, we address the design of our solution including the construct framework and notation visualisation. Section V then details the development process of our platform. In Section VI, we provide a brief overview of the implemented platform, including a run through of how security notation is added to an existing SIoT diagram. The evaluation of the proposed security notations then provided in Section VII. Section VIII analyse some existing BPMN security extensions identifying the notational issues

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment.

B. Zhou, C. Maines, S. Tang, Q. Shi, P. Yang. and J. Qi are with the Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, U.K (email: b.zhou@ljmu.ac.uk).

Corresponding author Q. Yang is with the College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China. (email: qyang@zju.edu.cn)

within each one. Finally the paper concluded with future work in Section IX.

II. Background

A. Use BPMN for SIoT Scenario Modelling

BPMN can be used to graphically represent business processes and their component relationships in a common standard between organisations [6, 7]. Apart from automatic service tasks that can be used to represent smart things, BPMN also provides the necessary social aspects such as the modelling of human tasks to demonstrate human activities. It allows interactions between humans and things to be captured in a single diagram. The Aniketo Project [8] has successfully utilised the BPMN platform to model a service-oriented environment incorporating both automatic services and manual user tasks. In our previous work [9], we also demonstrated the possibility to use BPMN diagrams to represent the IoT environment. In general, if we see each IoT device as a service provider, they can be represented in the BPMN diagrams as a service task. The whole scenario can then be described as a set of linked tasks, which must be executed in a specific order, collectively resulting in an objective or policy goal being achieved. These tasks can even be conducted across one or multiple organisations [10].

BPMN fulfils the requirement of visually representing process and is now the industry standard for business process modelling [11]. Nevertheless, even though security directly affects the functionality of these processes, BPMN has no support for specifying cyber security requirements [12, 13]. As it will be explained in Section VIII, current BPMN security extensions have made attempts, but they are being constructed unsystematically, without any empirical evidence to support their choice of concepts [14] or notational design.

B. Cyber Security Ontology

As a prerequisite, an ontology of cyber security requirements within the SIoT environment should be created first [5]. This allows ontological analysis to be conducted both throughout and after the creation of any BPMN security extension. Ontological analysis is a systematic process which prevents the issues such as construct deficit and ensures all necessary concepts are included. The process involves the comparison of a modelling language and an ontology to establish a one-to-one mapping between the two; either through interpretation mapping or representation mapping [5]. The ontology acts as a concept requirement list for the extension with ontological analysis determining whether or not all requirements are met.

For the cyber security domain such an ontology already exists. In our previous work, we presented an ontology of cyber security requirements that aims to act as a foundation for the creation of a comprehensive BPMN security extension [4]. Stressing that current extensions have been heavily construct deficit and thereby fail to adequately provide a suitable tool for representing security requirements, we proposed a total of 79 cyber security requirements that should be modellable in BPMN before an extension can be deemed comprehensive. These are structured into six areas with a hierarchy depth of four. The six areas being: *access control, privacy, availability, integrity, accountability, attack/harm detection and prevention* and *availability*. Full details of the ontology will not be discussed here as they can be found in paper [4].

With an ontology created, the next steps are to design and implement our solution. As mentioned, there are flaws with existing security extensions which are further elaborated upon in Section VIII. Throughout this paper we explore the hypothesis that a third dimension could provide the potential solution to these issues. By third dimension, we mean utilising 3D game technology to turn the conventional 2D-based BPMN diagrams into a 3D environment, with the extra dimension being used to represent security information visually. It potentially will help the users to understand and manage information more efficiently, especially in a complex scenario as explained later in the Section III.B. We propose a novel solution that aims to be comprehensive to the ontology created by us and satisfy Moody's "Physics of Notations" [5], which defines the principles a notational modelling language should stick with. We aim to use this ontology as a basis for our solution of the platform, ensuring that we are not only overcoming the notational issues but also providing the first truly comprehensive tool for modelling cyber security requirements within SIoT.

III. Modelling Approaches

A. Conventional Extension

Although there are many tools available for modelling (Visio, UML Designer, Modelio etc.), they are all very similar in functionality and user interaction. Take Microsoft's Visio as an example [15], see Fig. 1. The general user interface (UI) tends to include some form of workspace in the center of the screen. Typically on the left side of the UI, there will be a toolbar which includes the languages constructs. These are usually presented as their diagrammatic visual representation along with accompanying textual names. These constructs can then be added to the diagram through either drag-and-drop or by first selecting a construct then clicking somewhere within the workspace to place it there; this functionality being dependent on the software.

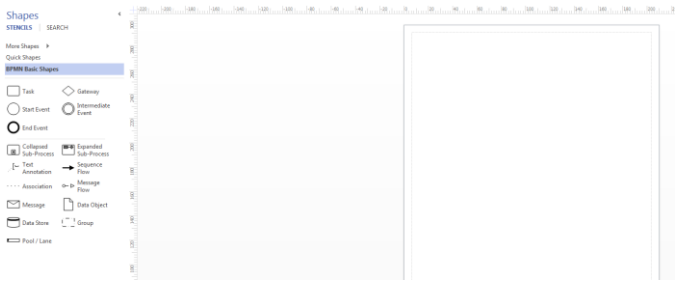

Fig. 1. Microsoft Visio [15]

There are some tools such as Activiti [16], which also feature detail bars along the bottom of the screen. These usually offer a more practical and familiar method of changing construct text or unique identifiers through text input boxes.

When developing a security extension, most authors will typically expand on an already existing tool such as Visio. Most tools usually include some functionality for the creation or inclusion of custom notation. SecBPMN2 for example [11], utilises Visio's Stencil functionality. This allows users to create a custom toolbar of their favourite symbols for use later on. It also allows for the saving of custom symbols as seen in Fig. 2.

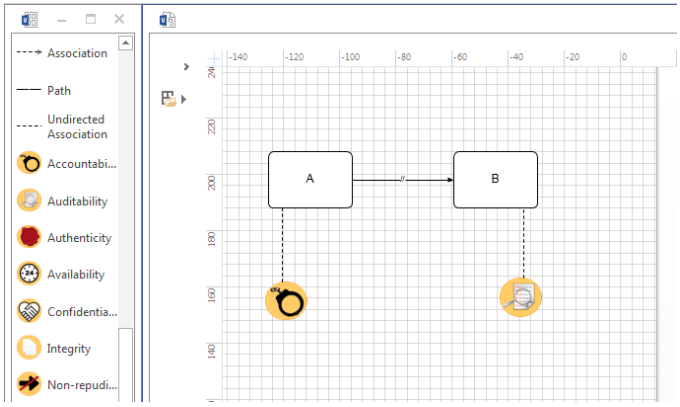

Fig. 2. SecBPMN2 - Visio Stencil

As most security extensions are represented in a similar manner to that of BPMN, there has been little progress in terms of new modelling approaches. Most authors merely present a set of concepts and accompanying symbols, regardless of the fact that by adding more constructs they have increased the languages complexity and potentially nullified any complexity management the language had in place.

B. *2D versus 3D*

To address the issue of complexity management, we propose representing cyber security requirements across the third dimension.

Research into 3D visualisations versus 2D visualisations has already provided empirical evidence supporting the use of 3D from both an efficient and user preference point of view [17]. The use of 3D in BPMN has also been investigated. The application created by Brown et al. [18] however does not feature any information across a third axis and merely provides a way of traversing or manipulating a 2D diagram. The advantage of our approach is that original SIoT scenario can remain relatively unaltered, being represented without change across two axes as seen in the left of Fig. 3.

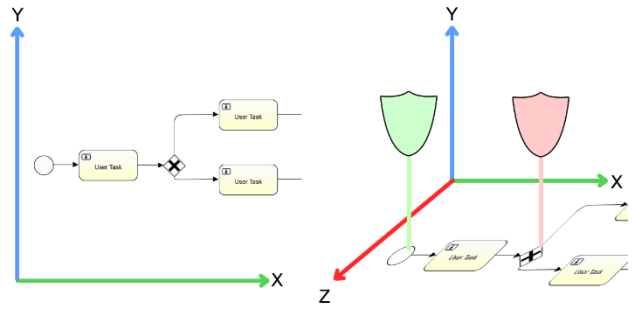

Fig. 3. 2D and 3D BPMN diagram

Cyber security requirements can then be represented across the *x*, *y*, and *z* axes, as seen in the right of Fig. 3. This way, cyber security requirements are being displayed at a similar abstraction level as BPMN whilst still maintaining a comprehensible diagram. The application of 3D however provides its own issues such as interactivity and rotation techniques [19]. While there is no objective to solve the problem in this study, it is worth noting what current or new means of interaction will be needed.

In another paper, we tested the feasibility of this hypothesis with user experimentation [20]. Participants were shown a simple BPMN diagram with several coloured circles linking to BPMN-task elements. Participants were then given a set amount of time in which they had to identify what colours were linked to each task, noting their results through a paper based questionnaire. The BPMN-task elements were assigned a number opposed to a business process, just as security was represented through coloured circles and not explicit notation. The intention being to represent current cyber security extensions in an abstract complexity-focused manner and not cause confusion or interference by including unnecessary business or security details.

From these experiments, we found that for BPMN diagrams with a relatively low number of security notation (six constructs), 3D provided no advantage in terms of read speed and accuracy. With most users preferring the 2D approach over 3D. However, when the complexity of the diagrams was increased (36 security constructs), 3D provided a substantial improvement in read speed and accuracy compared to 2D. More specifically, participants were able to read over 20% more of the 3D diagram with over twice the accuracy compared to 2D in the same amount of time.

IV. Designing a 3D BPMN Security Extension

Given the encouraging results gained from experimentation, the application of 3D to BPMN holds a lot of potential for representing cyber security requirements. Before detailing the visualisation of the language however, there must first be a notation to extend with.

A. *Notation – Visual Vocabulary*

By using parts of the "Physics of Notations" [5], we were able to create a graphical framework that can be used to create security constructs for every concept within our ontology [4].

Perceptual Discriminability: When extending an existing language with a new domain, perceptual discriminability has two sides. The notation not only requires discriminability amongst its own constructs, but also against that of the extended language – i.e. all notations representing IoT scenario are clearly distinguishable from security notations.

By representing security across the third dimension, BPMN and security notations will always be distinguishable first and foremost by whether or not they are across the *x/z* axes or the *x/y/z* axes.

Secondly, if a consistent shape is to be used for all constructs this means any distinguishing features must be encompassed within such a shape. A search on Google Images using the

keyword "*security*" [1] returned two most popular icons associated to security that we considered to use as outer shape in our design: *padlock* and *shield*.

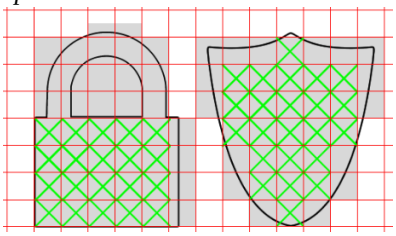

Fig. 4. Design space shape comparison

If one refers to Fig. 4 it shows that a padlock shape takes up roughly 40 squares. However, the inner design space only allows for 20 squares (in green), just 50% of the space the construct occupies. By comparison, the shield takes up 39 squares and allows for 23 full design spaces, i.e. 59% of the construct space. Therefore, in order to accommodate more discriminable features, we opted for a *shield* as the outer shape for our framework.

Semantic Transparency: The semantic transparency of the symbols will be achieved primarily through the creation of a unique icon. As the outer shape will be consistent (Fig. 5a), without anything inside the *shield* there is no way to distinguish the constructs from each other. Therefore, each construct will include an icon specific to the concept it represents (Fig. 5d). Using an icon design approach for symbols has been proven to increase usability, recognition and familiarity [21]. Inevitably every symbol will require some learning. However, using this approach will improve the semantic immediacy of the constructs given icons natural goal of being a graphical mnemonic to their concept.

Visual Expressiveness: There are eight visual variables can be used to construct a notation, these being: *horizontal position, vertical position, shape, brightness, size, orientation, colour* and *texture* [5]. We have already discussed how we have utilised *shape*, both as the outer shell of each construct and for the icons themselves.

We also utilised *brightness* within our notation as a way of inferring what hierarchy depth the symbol is at. The brighter the construct the higher the level, see Fig. 5e. However, as *brightness* is only a secondary notation, this hierarchy is reinforced with a more robust variable: *shape*. Viewing of Fig. 5b (and Fig. 5e) will show how we slightly changed the peaks of each shield to reiterate this hierarchy. We also used *size* to show this distinction. The higher the concept level the larger the construct. However, given that small symbols are difficult to read the *size* difference is relatively small. We deemed it more practical to keep the symbols at a readable size rather than utilise this variable to its full potential.

As for *colour*, this was used as a way of separating the six key areas as specified in our ontology, see Fig. 5f.

- Red = *access control*
- Orange = *privacy*

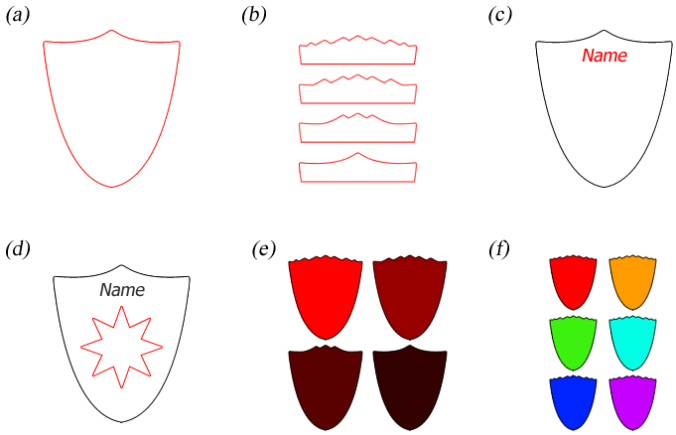

Fig. 5. Notation Construct Framework

- Green = *integrity*
- Turquoise = *accountability*
- Blue = *attack/harm detection and prevention*
- Purple = *availability*

Again, *colour* is only a secondary notation so this distinction is also reiterated elsewhere. In the next section we will discuss more about this.

Concentrating on just the notation design itself, the final variable used is *orientation*. As discussed earlier, security is always distinguishable by the fact it is perpendicular to the IoT diagram at a different *orientation*.

Dual Coding: Finally, dual coding. Given that this principle states that constructs should be accompanied by supporting text we added the name of each construct within the symbol. If you refer to Fig. 5c, you will see that our framework includes the concept name at the top of each shield.

An example of a full hierarchy of symbols can be seen in Fig. 6 (*size* is not demonstrated within this figure). This figure demonstrates the constructs: *access control* > *authentication* > *personnel authentication* > *biometric*.

Fig. 6. Symbol examples

From this you can see how *brightness* is used to determine the individual hierarchy level, along with the peak count at the top of each shield. It also acts as a good example of how simply following the "Physics of Notations" beforehand can drastically improve principles such as perceptual discriminability and semantic transparency. We do not claim to be expert icon designers. Nevertheless, by using other variables such as *shape*, *colour*, *brightness* and dual coding. The level of distinction amongst the symbols is a lot higher than the majority of the previously assessed extensions.

[1] Google Images, "Search Term: *security*," 2018, <goo.gl/Es7JcP> (accessed 30/01/2018)

B. Notation – Visual Grammar

Along with visual vocabulary (graphical symbols) a notation also requires visual grammar (compositional rules) [5]. For the proposed solution we build on from the 3D examples in our experiment [20]. That being, each IoT element will have its own unique holder capable of specifying any security requirements. However, representing 79 concepts at once on a single BPMN element will almost certainly cause cognitive overload and incur several complexity issues. Nevertheless, the incorporation of current BPMN functionality modularization [22], alongside 3D will allow for a more manageable and therefore comprehensible diagram.

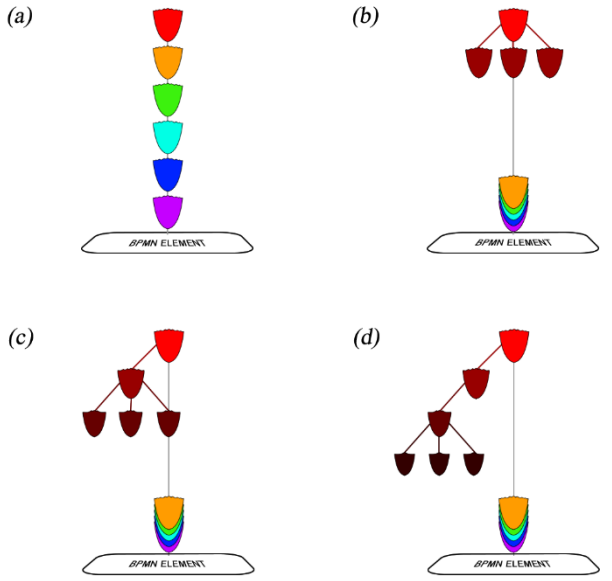

Fig. 7. Notation - Visual Grammar

For our solution we plan to display six concepts at the highest level on each IoT element (Fig. 7a), respective to the six key concepts identified by us [4]. These symbols will then act as individual buttons to modularise their sub-concepts. Once a symbol is selected the remaining five will collapse and the next level of concepts will display (Fig. 7b). This functionality will then continue for the lower levels (Fig. 7c-d). However, instead of collapsing the other symbols at lower levels, they will be hidden. This is to ensure complexity is still managed. Once collapsed the symbols become unidentifiable anyway, to hide them after the top level will ensure cognitive overload does not ensue.

To further iterate the concept hierarchy on top of *brightness*, *shape* and *size*, we include another visual variable: *vertical position*. Once a symbol is selected, the sub-concepts display at a decreased size respectively to the lowest level (Fig. 7d). They also appear below the parent symbol giving the impression of a tree structure and that a lower vertical position indicates a lower concept level. However, when no symbol is selected the six key concepts are displayed vertically as well. To ensure the user does not infer a similar hierarchy, new links (lines) are used to connect parent and child concepts (Fig. 7c-d). Along with the fact that the core six concepts also have different colours and the same size (not reduced sizes like their children), we are confident this issue will not arise. Nevertheless, we also colour the links (lines) respective to their core area to reiterate this relationship.

Along with *vertical position* we also use *horizontal position*. Unlike the majority of extensions (and modelling languages), our extension always places constructs in the exact same position relative to its associated IoT element explicitly. For example, if we were number each of the red symbols in Fig. 7d, one to six respectively. Symbol six (bottom right) will always appear in the same position irrelevant of whether or not symbols four and five are specified. This effectively means that if every icon was exactly the same, as they all have a unique position a user could still infer the construct from this alone.

Of the eight visual variables our extension utilises seven of them, dismissing only *texture*. Thereby making our solution one of the first to explore and utilise the full toolset at a modelling languages disposal.

C. Modelling Tool Functionality

Given that our proposed solution will include 3D visualisation, navigation of the scene is a potentially problematic area [19]. Nevertheless, we propose using a similar method as that used in most 3D games and game engines. Wherein, the user can 'fly' a camera around the scene using WSAD keys and mouse input configuration [23]. However, this may not be all users' preferences and some users will inevitably struggle to use this setup. Therefore, we will also include functionality to allow the user to align the camera to specific IoT and security elements with the push of a button.

One requirement to be considered is coherence. Existing extensions at times have a difference of opinion on what a concept is and thereby how it should be used. To overcome this issue, extensions would benefit from explicitly specifying concept meanings and consequently their use. To implement this into our solution, we included a details toolbar. This will allow the user to access various information about symbols which would otherwise be difficult to represent through a notation; such as the definition of a concept.

Where our ontology might be comprehensive to what a security notation should graphically represent, there is still room for more detail to be specified. For example, representing the security requirement for *virtual private network* in some cases may be adequate. However, there is more detail that could be specified such as L2TP or IKE [24]. Providing the functionality to specify such detail gives the modeller even more freedom to explicitly define their requirements. We represent such detail as text in a toolbar opposed to a graphical symbol to ensure graphic economy is maintained. When dropping to a level lower than the lowest of the ontology, the number of concepts hits an exponential increase. Take *biometric* as an example. If we were to include a graphical construct for each of *biometric's* children concepts. The graphic complexity of the extension could increase by around five (*iris*, *retina*, *facial*, *fingerprint* and *voice*) just from this one concept. Therefore, it is more appropriate to specify very specific detail as text instead.

D. Choice of Technology

When it comes to interactive digitally generated real-time 3D scenes projected onto 2D displays, game technology provides a range of technologies that support and simplify the development of interactive visualisation. Technical advancements in game technology include rendering, realistic physics, lighting, audio, graphical user interfaces (GUI), head-up displays (HUD), inputs and scripting [25]. They are also able to serialize and deserialize XML files.

The majority of engines available offer relatively similar functionality. The main differences being licensing costs and final render quality. If making a realistic first person shooter (FPS) for example, one of the best engines would be Unreal 4 [26]. However, taking into consideration what we require from an engine, we chose to work with Unity2. This is mainly due to Unity's functionality of being able to port to the majority of devices [27]. Although we have no immediate goal of developing on a platform beyond PC, this functionality is the most appealing to us in long term when compared to what other engines offered. Unity is also a free software to download.

E. Application Framework

We use two steps approach to create our BPMN diagram and include security. An overview of how the application works can be seen in Fig. 8.

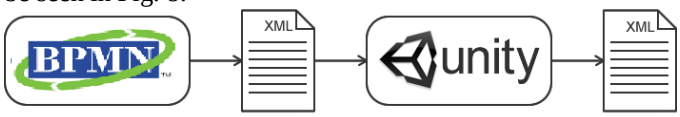

Fig. 8. Workflow Overview

Although a crude overview, this figure provides a visual representation of:

1. Creating a BPMN-based IoT diagram
2. Saving an XML file of the diagram
3. Reading the diagram into Unity
4. Adding security requirements to the diagram in Unity.
5. Saving an XML file of the security requirements.

The BPMN-based XML file contains all the necessary data to redraw its respective IoT diagram within a game engine environment. Each IoT element is represented by a node within the file specifying the following details: *element type*, *unique ID*, *height*, *width*, *x* and *y*. From this data we can assign a 3D model to each element respective to its *element type*. Then using the *width* and *height* attributes rescale this model as required. The *x* and *y* coordinates however must be remapped within the game engine to the *x* and *z* axes respectively. Where *x* and *y* are commonly used for 2D, the third dimension (*z*) is used to add depth. Therefore to make the diagram more readable and save awkward rotation within the engine, the coordinates need remapping (Refer to Fig. 3 to better understand this).

As previously mentioned, we plan to draw each security construct in the exact same location relative to its parent IoT element. This not only makes for a better notation, it also streamlines the development process. Referring back to Fig. 7, you will notice each IoT element has a grey line to show the linking of its security requirements; we call this a *holder*. As each construct will be in the same location respective to its IoT element. That means it will also be in the same location respective to this *holder*. As such, when adding security to each IoT element we are not required to store any coordinates. We simply need to specify what the associated IoT element is. From this, we can get the coordinates of that element and do a simple calculation to determine the *holder* location (we assume *y* is 0/ground level), see Fig. 9:

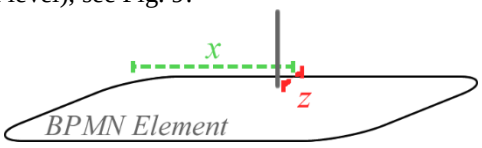

Fig. 9. Holder coordinates

$$x = x_2 + \left(\frac{w}{2}\right) \qquad (1)$$

$$z = z_2 + o$$

Where…

- *x = holder x coordinate*
- x_2 *= IoT element x coordinate*
- *w = width of IoT element*
- *z = holder z coordinate*
- z_2 *= IoT element z coordinate*
- *…and o is an offset value from the top of the IoT element.*

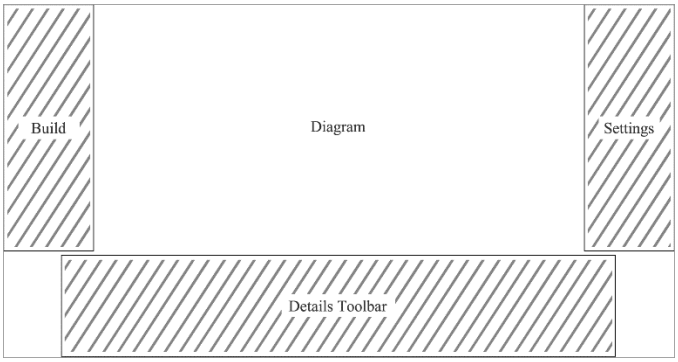

Fig. 10. Modelling Tool User Interface Wireframe

The Graphical User Interface (GUI) for our application will be split into three areas: *build*, *settings* and *details*. The *build* toolbar will consist of all the functionality required to add (and remove) security requirements to the IoT diagram. *Settings* as implied will control the various settings of the application, consisting of: hiding/showing of security requirements, focusing on a specific IoT element or security symbol and saving of the diagram itself. The details toolbar as previously discussed contains the various concept specifics (definitions

[2] Unity Technologies, "Unity:Home" - <https://unity3d.com/>, (accessed 30/01/2018)

etc.). Each area will be contained as seen in Fig. 10.

This reflects the conventional layout of modelling tools (see Section III) and should ensure a usable application through familiarity.

V. Development of a 3D BPMN Security Extension

The implementation of our application can be split into three phases: *planning*, *asset production* and *development*, see Fig.11.

A. Planning

The majority of the planning phase was covered earlier in the paper in Section II-IV. Specifically, defining what concepts we plan to include within our notation; those being the security requirements specified in our ontology [4].

As for the scope of supported IoT element, BPMN has a graphic complexity of 171 constructs [22]. As we are only testing a proposed extension not building industry-ready software, considering time constraints, we will only include support for a small portion of the BPMN notation. This being: *pool, lane, start event, end event, message start event, message catch event, timer start event, error end event, terminate end event, parallel gateway, exclusive gateway, inclusive gateway, user task, business rule task, script task, receive task*, and *service task*. Nevertheless, we are confident we have included sufficient notation for the coverage of most general SIoT scenarios.

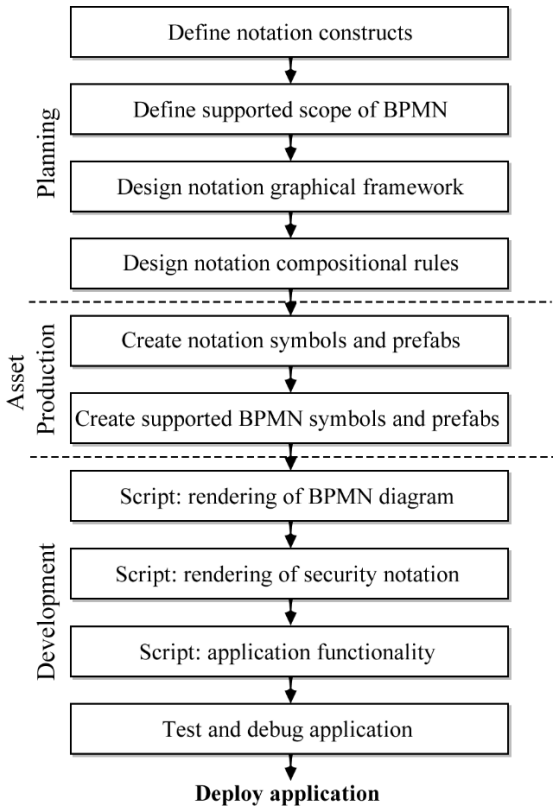

Fig. 11. Application development phases

B. Asset Production

When using an object (security construct) more than once in a game engine it is a good practice to assign that object a prefab. This allows all instances of the object to be edited at the same time and for the object to be easily instantiated from code [27]. To develop a prefab of a security notation requires just two components: a *3D plane* and a *diffuse texture*. We apply the diffuse texture to the plane. Then, so long as the background of our diffuse texture is alpha, we can toggle the setting within Unity to read all alpha channels as transparency. This game object can then be set to act as a prefab for the respective construct, as seen in Fig. 12.

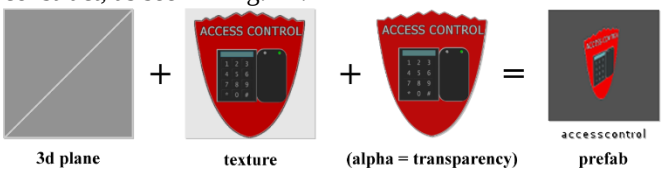

Fig. 12. Creation of construct prefab

The same process is also used to create the prefabs of supported IoT elements.

C. Development

The first stage of the development process was to render the IoT diagram. As each IoT element now has a designated prefab we could use the method described previously to redraw the diagram. Fig. 13 shows a screenshot of a rendered IoT diagram within the game engine environment.

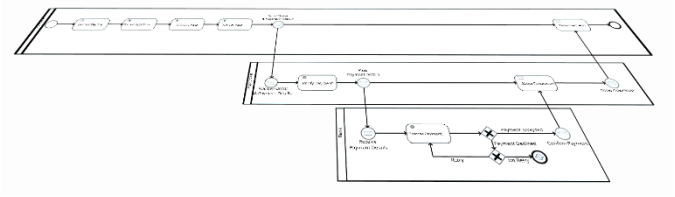

Fig. 13. IoT diagram rendered within game engine

The next stage involved parallel development of both: visualising security notation and functionality to add security notation to each IoT element. Within the application, security constructs are specified by:

1. Highlighting a IoT element (clicking on it).
2. Adding a holder to the element (button press).
3. Choosing a respective security construct (button press).

When a user assigns a security construct, rather than manually placing it like in other languages, the application performs some calculations and places the construct in a specific position relative to the IoT element and *holder*.

In the previous section we discussed how holders calculate their coordinates based on the associated IoT element. Each security construct has a similar relationship to its *holder*. From the *holder* coordinates, a security construct can use an equation respective to its position to calculate its own coordinates. For example, *access control* which is always at the top of the *holder* is calculated by first assigning the same coordinates as the *holder*. Then translating the construct *400 units* along the *y* axis and slightly offsetting the *x* coordinates to account for the symbol width, see Fig. 14. Similar equations are predefined within the application for every construct.

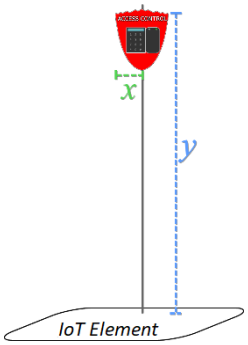

Fig. 14. Security construct coordinates

As for the modularisation of security constructs. This was accomplished by assigning each construct its own family tree. Should *access control* be selected, any construct whose parent is *access control* will then display. Similarly, if a child construct is highlighted, along with its own children displaying any siblings will collapse or hide depending on their level.

The details toolbar is another core functionality within our application. As stated early, coherence is an issue most extensions suffer from. Therefore, to overcome this issue, we included a toolbar that shows the definition of the currently highlighted concept, see Fig. 15.

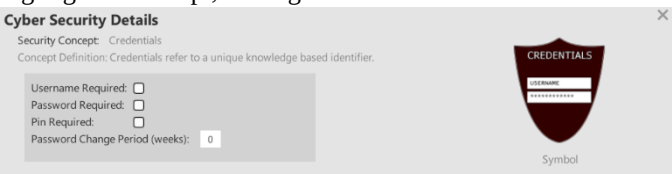

Fig. 15. Details toolbar - Credentials

Along with the concept definition, you will also notice in Fig. 15, the inclusion of lower level security detail. In this instance: *username required*, *password required*, *pin required*, and *password change period (weeks)*. As with the other security requirements in our extension, not all users will need or use such detail. Nevertheless, providing the functionality to specify this detail ensures an overall more comprehensive extension.

VI. Example Walkthrough

In order to explain how the application actually works. this section covers a more comprehensive walkthrough of adding security notation to an IoT diagram.

First of all, as a modeller you must decide on which IoT element you wish to add security to (all supported notations can specify security). Then using the left mouse button select the element as seen in Fig. 16.

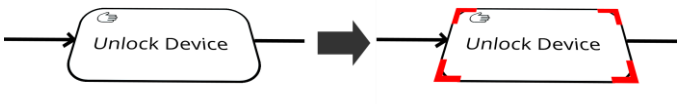

Fig. 16. Highlighting an IoT element

The application notifies the user that the IoT element is highlighted by placing an open-rectangle *highlighter* around it. It is also coloured red to further iterate this.

The next step is to add a *holder* to the element. This is accomplished by pressing the respective button in the *build* toolbar, as seen in Fig. 17.

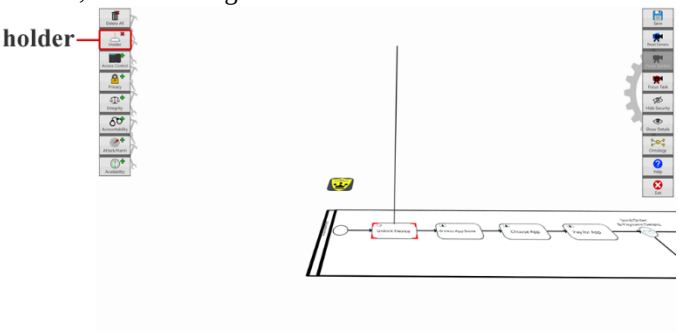

Fig. 17. Holder button location

Once an IoT element has a *holder* the application will give the user access to the six key concept as identified in paper [4] (Until a *holder* is added, this functionality is disabled.) The user then simply needs to add their choice of construct by pressing the respective button. These buttons are located under the build menu and can be seen to the left of the screen in Fig. 17.

However, this only works for the key six concepts. If the user wishes to add a concept of a lower level they must first select the parent construct by pressing it with the left mouse button. This will then change the set of constructs in the build menu to the children of the highlighted construct. Taking *access control* as an example parent. If we highlight this construct, the security requirements in the build menu will change from the core six to *authentication*, *identification*, and *authorisation*. This is another way of managing the complexity of the application along with the usability and speed at which a modeller can specify their requirements. When familiar with the notation hierarchy a modeller will be able to quickly add their requirements by selecting parent and child constructs. Compared with traditional methods of long scrollable lists of constructs our approach is much more manageable.

When selecting a parent symbol, the other symbols on the same level are either collapsed or hidden. An example of this can be seen in Fig. 18.

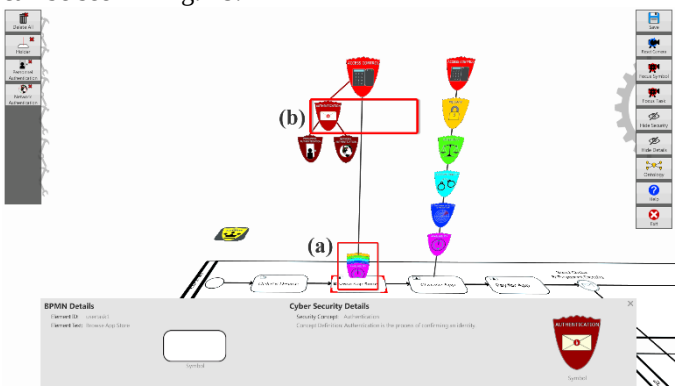

Fig. 18. Symbol hiding and collapsing

In this example, the highlighted concept is authentication, located inside Fig. 18b's red box. As the parent must also be highlighted (*access control*) the remaining concepts in the key six will collapse (Fig. 18a) (Refer to the element in the right side of the figure for an example of the key six not collapsed).

We will also be able to see by viewing Fig. 18b, how level two and lower concepts in the hierarchy hide their siblings when highlighted opposed to collapsing them. *Authentication* is siblings with *identification* and *authorisation*. However, as *authentication* has been highlighted in this instance, *identification* and *authorisation* are hidden from view. This is done as a way of managing the complexity of a diagram and reducing the visual clutter when trying to view a specific concept and its children.

As mentioned earlier, when a modeller is familiar with the notation hierarchy they will be able to work more efficiently when adding security requirements. However, it is unrealistic on our part to assume all users can only take advantage when familiar with the notation. Therefore, we included a button within the application which shows the ontology [4] when pressed. This way, users don't have to refer to external sources should they need reminding of a certain hierarchy or where a construct is located. The ontology within the application can be seen in Fig. 19.

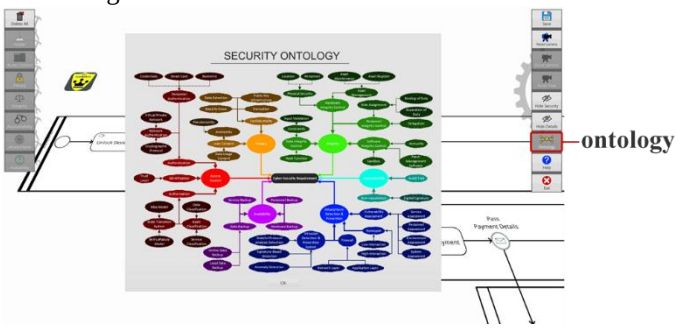

Fig. 19. Security ontology within application

For most constructs, specifying a link to an IoT element through parent/child links and *holders* is sufficient. However, there are some security concepts (e.g. *separation of duty* and *binding of duty*) which require linking to multiple IoT elements.

Within our application we included this functionality through the use of modularisation and the *details* toolbar. The modularisation side was already implemented as discussed earlier in this section. The *details* toolbar contains the functionality to specify any other IoT elements the constructs require linking to. This was done by using a form-style dropdown menu. The menu takes each text/name value of every IoT element within the diagram then lists them for the user to choose from (assuming they haven't already been linked; these elements are removed). This system can be seen in Fig. 20.

Fig. 20. Binding of duty IoT element linking

Once a user links another IoT element, the link is visually represented by a line across the diagram. The only way a user can access this functionality however is if the constructs themselves are visible on screen. When the hierarchy is collapsed and *binding of duty* is hidden, the links are no longer visible. An example of how this looks within the application be seen in Fig. 21.

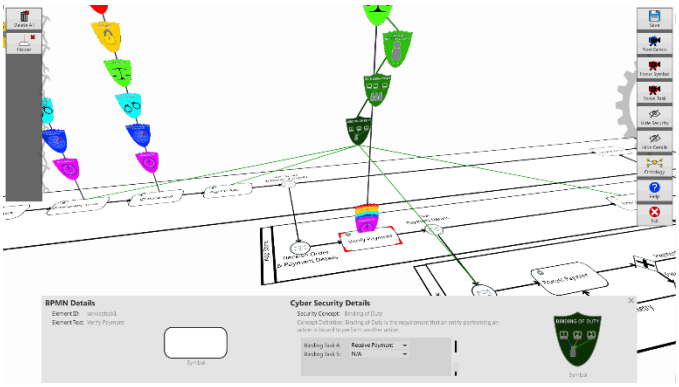

Fig. 21. Binding of duty diagram element linking

VII. Evaluation of the 3D Security Extension

We evaluate our extension against the "Physics of Notations" identified by Moody [5]. It defines nine principles used for evaluating any notations to ensure a scientific approach is kept in their assessment. These are design principles of notation not just for cyber security, but also for any notations used in real world such as traffic signs etc. They are frequently used in the evaluation of modelling languages as well including that of BPMN [22, 28].

A. Semiotic Clarity

Firstly, the semiotic clarity of the notation. Although some extensions appear comprehensive to the cyber security domain, this is only relative to the construct deficit many of the others suffer from. In fairness to several extensions, they do explicitly specify their focus on a particular area of security and as such their paucity can be excused. Nevertheless, needing multiple extensions to specify requirements for the same domain is very poor usability and will likely lead to the extensions being dismissed altogether. In our solution, semiotic clarity was guaranteed by ensuring every concept within our ontology has a respective construct within the notation.

B. Perceptual Discriminabiltiy

IoT and security notation are distinguishable instantly by the fact they are on different axes to each other. IoT is across the *x* and *z* axes, where security is across the *x*, *y* and *z* axes. All security notation is also encompassed by a *shield* shape. Whereas no IoT element uses this shape.

As for the security notation itself there are multiple factors that we have utilised to ensure perceptual discriminability. Every construct is in a unique position. Each of the six key areas have a unique colour and brightness dependent on hierarchy. All of the constructs have a unique icon -and finally- dual coding was used to label every construct in case all other variables fail.

C. Semantic Transparency

As for semantic transparency, we feel our notation falls

somewhere between transparent and translucent. Some concepts such as *firewall*, are very easily represented in a graphical form. Others such as *public key infrastructure* prove a lot more difficult. Nevertheless, we have created each icon with the intent of them being a mnemonic to their underlying semantics. As such, where a construct may not be semantically immediate. We believe with some definition and reasoning to how the icon was constructed, the user will be able to make the connection themselves in very little time.

Of course, each construct is also labelled, so after a few uses of the extension learnability alone will ensure semantic transparency.

D. Complexity Management

Given that complexity management was one of the core motivation for this project, we are certain our application meets this requirement.

To manage complexity we have split both notations (IoT and security) onto separate planes whilst still ensuring a relationship is maintained. We have also used modularisation to seamlessly tier security hierarchies and stop cognitive overload from occurring. This functionality was also incorporated into the modelling tool itself, whereby the application will only display functionality to add children of the currently highlighted construct. The modularisation also collapses and hides constructs from view that are not within the current hierarchy.

Given that our approach of modelling security has allowed for the inclusion of around 80 new constructs to IoT notations without cognitive overload, it proves how well our tool manages complexity.

E. Cognitive Intergration

Cognitive integration is a difficult principle to meet and in many ways requires a project such as this with a sole aim of solving it. In this instance this is not something we addressed. As it is not incorporated into IoT itself it is not something we prioritised to solve within this project.

F. Visual Expressiveness

As mentioned earlier in the paper, visual expressiveness was done very successfully within our notation. *Horizontal* and *vertical position* were used to show both hierarchy within the notation as well as to provide each construct a unique and identifiable location. *Brightness* and *size* (although *size* was discreetly used) were then utilised to further iterate this hierarchy.

Colour was used as a way of distinguishing between each of the six key areas as defined in our ontology. *Shape* was utilised for the both the holder shape (*shield*), and the unique icon of each construct. *Orientation* most importantly was used a way of distinguishing IoT and security notation. The only variable this application did not utilise was *texture*. By hitting seven of the possible eight visual variables however, we are confident in stating our notation as being visually expressive.

G. Dual Coding

Dual coding, a relatively straightforward principle to achieve can be seen by viewing any of our previous figures which include a notation construct. Within each *shield* we placed the name of the concept across the top.

H. Graphic Economy

Linking into complexity management, graphic economy ensures notations do not reach exceedingly high numbers of constructs. Along with issues such as construct redundancy, having a poor graphic economy can be overwhelming to a user, especially a novice [5].

Given the method in which we allow access to our constructs we believe that our notation has a good economy. If we had listed all of our constructs in a dropdown menu as most other tools do, our economy could be considered poor. Nevertheless, given that users are only ever subject to around six concepts or less at any given time, we are confident in stating this principle has been met.

I. Cognitive Fit

Cognitive fit is a principle we achieved by modularisation of the security hierarchy. By showing six concepts at the highest level acts as a mechanism for novice users to utilise the notation. As expertise and domain knowledge increases users can then select one of these six and specify more specific concepts. Therefore, we consider the application to have four levels of cognitive fit, respective to the four tiers of each security hierarchy.

In summary, of Moody's nine principles, our extension satisfies eight of them. Although some still have room for improvement, given that we created our notation very much with these principles in mind ensured we were able to satisfy most of them.

J. Experiments and Results

As previously mentioned, we carried out experiments to evaluate our 3D platform against conventional 2D solutions in paper [20]. Participants were asked to implement the same tasks within the same time period in 2D and 3D environments respectively. Their completion rate and accuracy were measured in comparison.

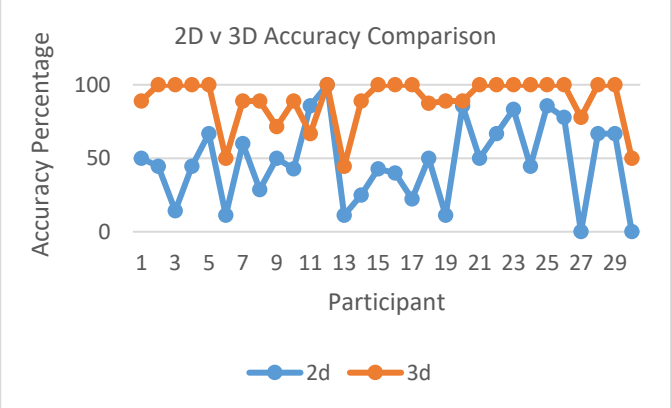

Fig. 22. 2D v 3D accuracy comparison

Fig. 22. illustrates the performance difference between the 2D and 3D approaches. With the exceptions of Participant 12 who achieved 100% accuracy in every experiment, and Participant 11 who performed better on the 2D solution. Every

other participant achieved a higher accuracy percentage when using 3D solution. Put into numbers, on average, participants improved their accuracy by 169.1% ±69.33% when working in 3D environment compared to working in 2D, although the margin of error is quite large in this instance. Even if we view the accuracy from a worst-case perspective in regards to this, the average participant still made a 100% accuracy improvement. The same can be said for the completion percentage, though not as substantially different, participants completed 21.05% ±8.18% more of the 3D diagram tasks compared with the 2D diagram, as shown in Fig. 23.

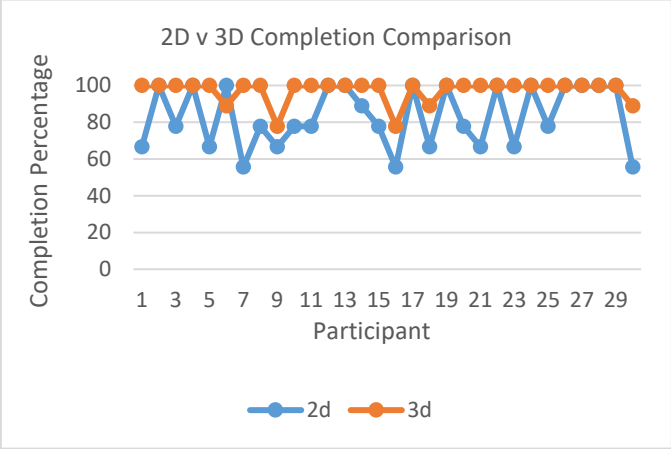

Fig. 23. 2D v 3D completion comparison

VIII. Related Work

There are some existing security extensions created for BPMN. We evaluate them by using the same principles defined by Moody.

The security notation shown in Fig. 24 is the extension created by Rodriguez et al. [12]. The symbols from left-to-right represent: *non-repudiation*, *attack harm detection*, *integrity*, *privacy*, and *access control*. Perceptual discriminability has both strengths and weaknesses in this extension. The symbols are very distinct from the BPMN notation; separating their domain well from business processes. However, they are far too similar to each other, with only a few textual characters being the difference. Given that the distinguishing factor is text - making international use extremely difficult- this extension fails to adequately meet this principle.

Fig. 24. Rodriguez et al. security notation

The semantic transparency of the symbols is of a similar nature. A padlock successfully infers the meaning of security, but as all the concepts are represented by a *padlock*, without the use of text any further detail is impossible to distinguish; failing this principle. This leads onto the visual expressiveness of the symbols. There are eight visual variables which can be used to construct a notation, these being: *horizontal position*, *vertical position*, *shape*, *brightness*, *size*, *orientation*, *colour*, and *texture* [5]. This extension is only utilising one of the eight; that being *shape*. Very little potential has been taken advantage of graphically.

The complexity management of this extension is non-existent having adopted the method of stamping symbols onto BPMN elements. Where this may be an effective way of linking BPMN tasks and security concepts, diagrams quickly become overwhelmed when multiple concepts are placed on a single element. In the paper that introduces it [12], the symbols are displayed on a BPMN diagram at their target size; when scaled, the text on each notation is very difficult to read. Given that text has been used as primary notation, it is extremely poor design given that once the symbols are displayed at a usable size the text becomes nearly unreadable. Ideally text should only ever be used as a secondary notation to reiterate a semantic meaning. It has very poor cognitive effectiveness and provides very little discriminability between symbols [5]. However, as it has very much been used as a core distinguishing variable in this case, it is by far the biggest weakness of the design. Further appreciation of this fault can be found when viewing the notation from a non-English speaking perspective.

The dual coding principle states that text, more specifically words, should be used to complement graphics [5]. Used correctly text can be a useful tool for learning a notation. A novice user for example can keep referring to construct labels until they are confident enough using just visual aids. However, using acronyms adds to the difficulty of this as now novices must not only learn graphical symbols but these as well. In this instance, dual coding has not been satisfied.

The graphic economy of this notation is fairly successful. Nevertheless, as mentioned earlier, there are concept omissions within the extension. A notation with poor graphic economy is more desirable than one with construct deficit. As for cognitive integration and cognitive fit, the paper made no mention of any functionality to support either of these principles.

Saleem et al. [13], slightly newer, security extension takes a different approach notation-wise, but still yields several issues. In Fig. 25 from left-to-right, the symbols represent *confidentiality*, *integrity*, and *availability* respectfully; the core concepts of cyber security [29].

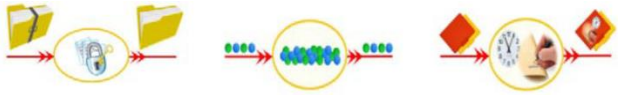

Fig. 25. Saleem et al. security notation

In this case, the perceptual discriminability of the symbols was done rather successfully. Although similar in theme, each notation has several distinguishing variables that don't include text. This makes the unique identification from any culture or language very easy. The same can be said about the semantic transparency. This extension has at least had some thought put into semiotics, creating a notation whose symbols perceptually resemble their semantic meaning. However, there are still some areas that allow for improvement. Take integrity for example; colour has been used as a primary notation to show an identical pattern both before and after a transmission; an appropriate visualisation. A better option however, would be to use shapes such as triangles and squares. *Colour* is very useful in notations, but like *text* should be a secondary notation not a primary notation [5]. Some people can struggle to distinguish between

certain colours, with the likes of blue and green being a prime example for people who suffer from Tritanopia [30] a variant of colour blindness.

The visual expressiveness of these symbols appears much higher than that of Rodriguez et al. with around three variables being used this time (*colour*, *texture*, and *shape*). However, visual expressiveness refers to the idea that multiple variables should be used as a form of secondary notation. The only variable distinguishing these symbols is *shape*; *colour* and *texture* are merely decoration. To better utilise *colour* for example, each symbol should have had its own unique one e.g. red, blue and green.

Complexity management is the same as with the previous extension (given the impact of this principle to visual notations it is a wonder so few acknowledge it). The paper that introduces this extension also provides a complete BPMN diagram incorporating the symbols. The notations again have very poor scalability; but not as bad as that of Rodriguez et al. as there is a much higher perceptual discriminability in these symbols. Nonetheless, these symbols are not enclosed in a uniform shape as with the previous extension, when close together their boundaries become hard to see and they begin to corrupt each other.

As seen in Fig. 25 the principle of dual coding has not been met. The graphic economy is similar to that of the previous extension. Although it may be economical, the reason of it being construct deficit is a worse anomaly. As for cognitive fit and cognitive integration, they were also neglected by Saleem et al.

Salnitri et al. SecBPMN2 [11] notation can be seen in Fig. 26. From left-to-right the concepts are as follows: *integrity*, *authenticity*, *accountability*, *non-repudiation*, *auditability*, *confidentiality*, *privacy*, *binding of duties*, *separation of duties*, *availability*, and *non-delegation*.

Fig. 26. SecBPMN2 security notation

The perceptual discriminability of these symbols is somewhat successful, but the visual distance between each symbol is dependent exclusively on *shape*. Nevertheless, as each symbol is clearly distinguishable from the others, the extension satisfies the principle. As for semantic transparency, these symbols fall somewhere in the semantically translucent range. They are not capable of semantic immediacy but nor are they are opaque. There has been some thought put into their design but there are still uncertainties as to their exact meaning; a weak satisfaction of the principle. The visual expressiveness of the symbols as touched on earlier is limited again; the only variable in use is *shape*. Some may argue colour has also been used, as mentioned earlier though this must be in the form of secondary notation. Nevertheless, given that orange has been used consistently throughout all the symbols, separates this notation well from BPMN (which tends to be black and white or pale pastel colours).

The complexity management of this extension is rather poor. Viewing it in use in Fig. 27, it is easy to see how some users may feel in a state of cognitive overload. This is partially due to the dashed lines across the diagram, however given that very few concepts have been used, it is easy to see how this extension can quickly overwhelm a diagram.

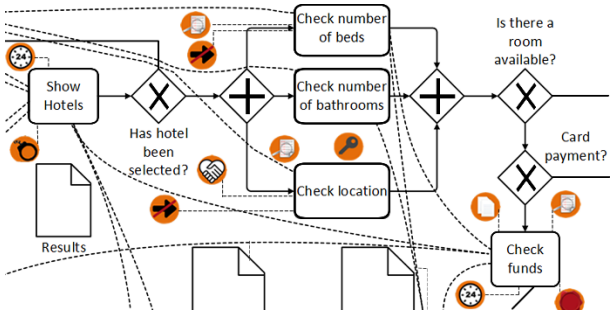

Fig. 27. SecBPMN2 business process model

The remaining principles follow a similar theme as the previous extensions. Graphic economy is achieved through a construct deficit language, dual coding is not included and cognitive fit and cognitive integration are altogether ignored.

Labda et al. security notation [31] can be seen in Fig. 28. The symbols from left-to-right represent: *access control (allow)*, *access control (prevent)*, *access control (limited)*, *separation of tasks*, *binding of tasks*, *user consent*, *necessity to know (high)*, *necessity to know (medium)*, and *necessity to know (low)*.

Fig. 28. Labda et al. security notation

The perceptual discriminability of this extension is difficult to determine. Each symbol is undoubtedly unique from the others but the three *necessity to know* constructs have little visual distance. There appears to have been an attempt at adhering to the principle of semantic transparency. Nevertheless, the symbols are borderline between semantically opaque and perverse. For example, the concept of *separation of tasks* is represented by a lightning bolt; a strange choice of mnemonic. Given that these symbols don't exactly infer their semantic meaning it is difficult to conclude semantic transparency has been met.

The visual expressiveness of the symbols is similar to the previous extensions. The only utilised variable is *shape* with *colour* once again acting as a weak secondary notation to separate the security notation from BPMN.

The complexity management of this extension is the same as Rodriguez et al. [12]; also using the symbol stamping method. As expected similar themes emerge. To place the concepts on BPMN elements they require scaling which once again makes perceptually discriminability difficult, especially given there's a visual distance of just one (*shape*). The graphic economy of this extension is controlled but this again is dependent on whether or not semiotic clarity was satisfied in the first instance. Cognitive fit and cognitive integration again are not acknowledged within this extension with dual coding clearly not included by observation of Fig. 28.

One of our previous notation [9] (see Fig. 29) is similar to

that of Salnitri et al but particularly with aim of IoT devices. Both opting for a circular shape with some form of icon inside. The underlying semantics for this notation are *security task*, *authentication*, *access control*, *authorisation*, *harm protection*, *encrypted message*, *non-repudiation*, and *secure communication* respectively left-to-right. The final three symbols represent *confidentiality*, *integrity*, and *availability*. With the stars visualising the required level for each as discussed in the previous section.

Fig. 29 Koh and Zhou security notation

Starting again with the perceptual discriminability of the notation, this extension has both positives and negatives. We used a padlock on each symbol as a way of identifying and separating them as security constructs. This is an effective way of separating the notation from BPMN. However, given that the general design is very similar to that of BPMN (black and white icons in circles) the notation isn't as semantically immediate as it could be. Nevertheless, the use of padlocks and the icons inside each shape ensures the satisfaction of this principle (although there is room for improvement).

The semantic transparency of these symbols is similar to Labda et al. notation. It is clear from inspection of each element that thought has gone into making each symbol. However, some of the symbols are semantically perverse. Take *access control* for example (third symbol along on the top row). This icon is the universal symbol for "shuffle mode" on audio devices. Although expert security users may have a different meaning, the majority of business and novice users will be more likely to associate this icon with "shuffle". A less generic icon should have been used.

The complexity management of this extension is a combination of symbol stamping and BPMN element replacement. By this we mean, rather than use a BPMN *message event*, one would use a security element *encrypted message*. However, we realised that this was a poor way to model security. Although business and security directly affect each other, readers do not always want or need to see both domains. Extensions should aim to be as non-intrusive as possible, allowing modellers the ability to remove (or hide) security requirements whilst still maintaining a complete business process (Effectively a complexity management system on its own). A portion of a BPMN diagram can be seen in Fig. 30 demonstrating this extension in use.

This figure emphasises our previous point on how the notation is far too similar to BPMN. It is not as clear in this figure as in SecBPMN2 what is security and what is business process. On further inspection it will be able to identify eight security elements in use. Nevertheless, there has been no attempt at managing the extra complexity created by the extension. Once again failing this principle.

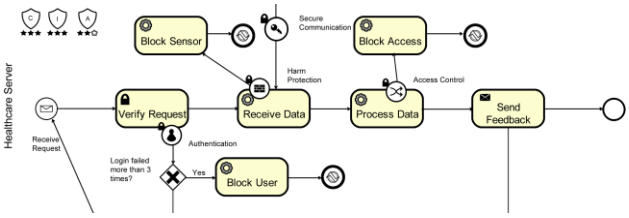

Fig. 30 Koh and Zhou business process model

The visual expressiveness of this extension is rather poor, utilising only one variable: *shape*. *Confidentiality*, *integrity*, and *availability* could be considered to use *horizontal* and *vertical position* also as they seem to consistently appear in the top left of each diagram in the authors' paper. However, it is not explicitly defined whether this is a rule of the notation.

Dual coding is also neglected within this extension. None of the elements feature any supporting text, with *confidentiality*, *integrity*, and *availability* once again using letters as discriminable features. The graphic economy carries on from the previous section, although the number of elements is manageable being construct deficit is worse. As for cognitive fit and cognitive integration these were also left out of this extension. For all the drawbacks we identified I use, it motivated us to look at the issue from the perspective of third dimension.

From this review, of the five extensions evaluated not one extension is capable of satisfying even half of Moody's principles. Moody discussed how there are trade-offs amongst the principles and satisfying one may have a negative effect on another. Nevertheless, certain principles such as complexity management should always be achieved, especially in software engineering [5]. In contrast, our security extension proposed in this paper is not only comprehensive to the domain but can also satisfies the most optimal number of principles.

IX. Conclusions and Future Work

Security is one of the key problems needs to be addressed before the wide application of IoT devices. The deep engagement of smart devices into our daily lives means a Social IoT environment comprises interactions between both humans and the things. While benefiting from the convenience it brought to us, the security issue could have even bigger impact and more severe consequences on our society [32, 33]. Understanding the security issue requires us to first capture and model the security requirements in such dynamic environment.

In this paper, we explored the possibility to use BPMN to represent SIoT scenarios. The nature of business process, which contains both automatic service task and manual user task means BPMN is capable of capturing both the technical and social aspects of SIoT environment. Based on this, we proposed to use the third dimension to model the security requirements. Unlike traditional 2D based solutions, which suffer from the complexity management issue, our solution separate the security notations from the IoT diagram in different dimension

thus makes it much easier to manage for even non-security experts. We designed the new 3D security extensions by following the design principles outlined by Moody for notations. It is comprehensive (supported by our ontology) and proving to be capable of satisfying the most number of these principles.

A complete framework for our notation (both symbolic and visual representation) was detailed with explicit note on how we achieve complexity management (amongst other principles). We then discussed how we implemented our proposed notation within a game engine focusing specifically on the functionality we created to ensure complexity management and usability.

As these principles are only a theoretical evaluation, in our future works we plan to test the application with real users. This way we can assess the application from a usability perspective as well as gauging the potential adoption of the application within an industry SIoT environment.

REFERENCES

[1] J. Zhang, et al., "Cyber-Physical-Social Systems: The State of the Art and Perspectives," *IEEE Trans. on Computational Social Systems*, vol. 5, issue 3, pp. 829–840, 2018.

[2] A. Paradise, A. Shabtai, R. Puzis, A. Elyashar, Y. Elovici, M. Roshandel, and C. Peylo, "Creation and Management of Social Network Honeypots for Detecting Targeted Cyber Attacks," *IEEE Trans. on Computational Social Systems*, vol. 4, issue 3, pp. 65–79, 2017.

[3] Y. Zhang, C Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure Certificateless Public Verification for Cloud-Based Cyber-Physical-Social Systems Against Malicious Auditors," *IEEE Trans. on* Computational Social Systems, vol. 2, issue 4, pp. 159–170, 2015.

[4] C. L. Maines, D. Llewellyn-Jones, S. Tang and B. Zhou, "A cyber security ontology for BPMN-security extensions," in *The 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015)*, Liverpool, UK, 26th-28th October, 2015.

[5] D. Moody, "The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756-779, 2009.

[6] P. Bocciarelli and A. D'Ambrogio, "A BPMN extension for modeling non functional properties of business processes," in *TMS-DEVS '11 Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, Boston, 4th-7th April, 2011.

[7] "Business Process Model and Notation," OMG, [Online]. Available: http://www.bpmn.org/. [Accessed 30 January 2018].

[8] A.D. Brucker, B. Zhou, F. Malmignati, Q. Shi & M. Merabti. "Modelling, validating, and ranking of secure service compositions." *Software: Practice and Experience*, vol. 47, issue 12, pp. 1923-43, December, 2017.

[9] S. S. Koh and B. Zhou, "BPMN security extensions for healthcare process," in *The 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015)*, Liverpool, 2015.

[10] M. Chinosi and A. Trombetta, "BPMN: An introduction to the standard," *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124-134, January 2012.

[11] M. Salnitri, F. Dalpiaz and P. Giorgini, "Modeling and Verifying Security Policies in Business Processes," *Enterprise, Business-Process and Information and Information Systems Modeling, Springer LCBIP*, vol. 175, pp. 200-214, 2014.

[12] A. Rodríguez, E. Fernández-Medina and M. Piattini, "A BPMN extension for the modeling of security requirements in business processes," *IEICE Transactions on Information and Systems*, Vols. E90-D, no. 4, pp. 745-752, April 2007.

[13] M. Q. Saleem, J. B. Jaafar and M. F. Hassan, "A Domain-Specific Language for Modelling Security Objectives in a Business Process Models of SOA Applications," *Advances in Information Sciences and Service Sciences*, vol. 4, no. January, pp. 353-362, January 2012.

[14] M. Leitner, M. Miller and S. Rinderle-Ma, "An Analysis and Evaluation of Security Aspects in the Business Process Model and Notation," in *International Conference on Availability, Reliability and Security*, Regensburg, 2nd-6th September, 2013.

[15] "Visio Home," Microsoft, [Online]. Available: https://products.office.com/en-gb/visio/flowchart-software. [Accessed 30 January 2018].

[16] "Activiti BPM Platform," Alfresco, [Online]. Available: http://activiti.org/. [Accessed 30 January 2018].

[17] F. Amini, S. Rufiange, Z. Hossain, Q. Ventura, P. Irani and M. J. McGuffin, "The Impact of Interactivity on Comprehending 2D and 3D Visualizations of Movement Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 1, pp. 122-135, January 2015.

[18] R. A. Brown, J. C. Recker and S. West, "Using virtual worlds for collaborative business process modeling," *Journal of Business Process Management*, vol. 17, no. 3, pp. 546-564, 2011.

[19] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt and N. Kassell, "Usability Analysis of 3D Rotation Techniques," in *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology - UIST '97*, Banff, pp. 1-10, 14th - 17th October, 1997.

[20] C. L. Maines, B. Zhou, S. Tang and Q. Shi, "Adding a Third Dimension to BPMN as a means of Representing Cyber Security Requirements," in *9th International Conference on Development in e-Systems Engineering (DeSE2016)*, Liverpool, 2016.

[21] L. Kascak, N. Ave, C. B. Rébola and J. a. Sanford, "Icon Design for User Interface of Remote Patient," in *SIGDOC '13 Proceedings of the 31st ACM International Conference on Design of Communication*, New York, 2013.

[22] N. Genon, P. Heymans and D. Amyot, "Analysing the Cognitive Effectiveness of the BPMN 2 . 0 Visual Notation," *Software Language Engineering, Springer LNCS*, pp. 377-396, 2010.

[23] "Scene view navigation: Unity Manual," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/SceneViewNavigation.html. [Accessed 30 January 2018].

[24] T. Sharma and R. Yadav, "Security in Virtual private network," *International Journal of Innovations & Advancements in Computer Science*, vol. 4, no. March, pp. 669-675, 2015.

[25] C. Maines and S. Tang, "An Application of Game Technology to Virtual University Campus Touring and Interior Navigation," in *7th International Conference on Development in e-Systems Engineering (DESE2014)*, Phapos, 25th - 27th August, 2014.

[26] J. Gregory, Game Engine Architecture, CRC Press, 2014.

[27] "Prefabs: Unity Manual," Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/Prefabs.html. [Accessed 30 January 2018].

[28] G. Popescu and A. Wegmann, "Using the Physics of Notations Theory to Evaluate the Visual Notation of SEAM," in *16th IEEE Conference on Business Informatics (CBI)*, Geneva, pp. 166-173, 14th-17th July, 2014.

[29] C. P. Pfleeger and S. L. Pfleeger, Security in Computing, Prentive Hall PTR, 4th Edition, 2006.

[30] "Tritanopia – Blue-Yellow Color Blindness," Colblindor, [Online]. Available: http://www.color-blindness.com/tritanopia-blue-yellow-color-blindness. [Accessed 30 January 2018].

[31] W. Labda, N. Mehandjiev and P. Sampaio, "Modeling of Privacy-Aware Business Processes in BPMN to Protect Personal Data," in *29th ACM Symposium on Applied Computing*, Gyeongju, pp. 1399-1405, 24th - 28th March, 2014.

[32] Y. Qu, S. Yu, L. Gao, W. Zhou, and S. Peng, "A Hybrid Privacy Protection Scheme in Cyber-Physical Social Networks," IEEE Trans. on Computational Social Systems, vol. 5, issue 3, pp. 773–784, 2018.

[33] P. Chattopadhyay, L. Wang, and Y. Tan, "Scenario-Based Insider Threat Detection From Cyber Activities," *IEEE Trans. on Computational Social Systems*, vol. 5, issue 3, pp. 660–675, 2018.