Abstract—Requirements engineering (RE) involves processing informal natural language descriptions of system requirements into an integrated and structured formal specification. The traditional RE process is ad-hoc, laborious, time-consuming, and error-prone manual process. For large-scale software intensive systems, the RE process becomes very difficult to manage. It is further complicated by ill-defined, incomplete, and redundantly specified requirements. To streamline the process, an end-to-end seamless RE framework is therefore needed. In this paper, we present an overview of an end-to-end semi-automated change-tolerant interactive requirements engineering (iRE) framework. It keeps and maintains the meta-level information of the given requirements models and their interrelations into a semantic network model (SNM), and provides a method and processing system to derive an integrated and structured model. The iRE involves the requirements analysts in an interactive fashion during the modeling process.

Index Terms—Requirements Engineering, Behavior Engineering, Behavior Trees, Semantic Network Model, Defects Detection, Change Management, Model Integration.

I. INTRODUCTION

Requirements Engineering (RE) often deals with a set of scattered and unstructured system requirements with a view to derive an integrated and structured system model that would serve as a formal specification. The requirements are usually represented by graphical models. The requirements analyst (RA) manually visits every model along with its interrelated models, and perform sanity and integrity checking. In this process, the RA, temporarily or finally, makes decisions on which requirement models are at that moment ready to be part of the formal specification. This decision may affect decisions made for other interrelated models in a cascading fashion.

Nevertheless, the traditional RE process is ad-hoc, laborious, time-consuming, and error-prone manual process. It is further complicated by ill-defined, incomplete, and redundantly specified requirements. Sometimes new requirements may be introduced, while existing requirements may be modified or deleted. Since the requirements are interrelated, a change in a requirement affects the acceptability of its related requirements for inclusion in the specification; which may again have ripple or cascading effects on other requirements further. Putting all these together, the whole process becomes very difficult to manage for a large-scale software intensive system, even if it is somewhat manageable for a small-scale one. To streamline the process, an end-to-end RE framework is therefore needed.

In this paper, we present an overview of a new end-to-end semi-automated change-tolerant interactive requirements engineering (iRE) framework. It keeps and maintains the meta-level information of the given requirements models and their interrelations into a semantic network model (SNM), and provides a method and processing system to derive an integrated and structured model. The iRE involves the RA in an interactive fashion in the whole modeling process. We also have developed a prototype GUI toolkit [1] for the behavior trees (BTs), a modeling language used in Behavior Engineering ([2]–[4]). BT notation can be used in the entire RE process. We have not chosen other modeling languages such as the Unified Modeling Language (UML) because they pose additional significant challenges due to their use of a number of different models (e.g. use cases, sequence diagrams, state charts, etc.) in the process, and thus making the formalisation of an end-to-end seamless RE framework difficult.

Paper organisation: Section II describes the background about RE process using BTs; Section III briefly describes our interactive RE framework; Section IV illustrates the iRE; Section V describes our GUI toolkit; Section VI explores related literature; and finally Section VII presents our conclusion.

II. BACKGROUND

Consider the requirements of the Security Alarm System (SAS) ([5]) shown in Table I. The SAS will be used as a running example in this paper.

1) SAS: The SAS when activated by pressing a set button displays its active status. While active, the SAS can detect motion and can then make alarm sound. The alarm sound can be deactivated by entering a 3-digit code, which can be cleared and re-entered, if a mistake is made. From these requirements, an integrated and structured model would be developed.

<table>
<thead>
<tr>
<th>TABLE I REQUIREMENTS OF THE SECURITY ALARM SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
</tr>
<tr>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
</tr>
<tr>
<td>R7</td>
</tr>
</tbody>
</table>

Note that the SAS is not a representative of the scale of problems that we intend to deal with. The SAS is rather a carefully selected small problem that is interesting enough to elucidate our model and mechanisms.
In this paper, we represent the SAS using BTs i.e. behavior trees. BTs come very close to natural language specification in terms of what they can express [6]. They capture dynamic behaviors of a system in an arguably easy-to-understand graphical format in a manner which stays close to the structure and terminology of the original requirements, but possess a formal semantics [4]. We argue BTs can be used seamlessly throughout the modeling process since it combines use cases, scenarios and state-based behaviors into a single model. Moreover, state machines can be derived from a behavior tree [7].

2) SAS and BTs: Requirement R2 in Table I mentions the initialisation scenarios for activating the SAS. R2 is represented by a BT in Fig. 1(a) as BT2. Fig. 1(b) displays the full contents of a BT node. Each node is associated with (a) a component, which has a behavior described by (b) a behavior name and (c) a behavior type. The node has a tag consisting of (d) a status (e.g. original, implied (+), and missing (−)) indicating the level of modifications made apart from the original requirements, and (e) a traceability link to trace the node back to the original requirements. It can also have (f) an operator and/or (g) label which describe the flow of control. The node further may have (h) related nodes to describe relational behavior with other components. A relational behavior may be described by the related component name, the type of relation (what, how, where etc) and its qualifier (on, at, in etc). Note the brackets show the syntax.

![Behavior Tree and Attributes of a node](image1.png)

![Behavior Trees (dots: nodes were skipped)](image2.png)

Behavior types are internal (>..<) and external (?>..<?) inputs, internal (<?..>) and external (>..?>) outputs, event (?..??), selection, (?..?), state realisation (..?) etc (Fig. 3).

![Behavior Types of a node](image3.png)

The node operators are used to indicate control flow beyond simple sequential execution and branching. An operator (at source node) refers to another equivalent node (at target node) elsewhere in the behavior tree. The nodes of a behavior tree can be connected in different ways such as sequential, parallel, atomic and alternative ([3], [8]). Formal semantics have been provided to behavior trees using an extension of CSP which can capture state based information [4].

Notice BTs for Requirements R2 and R3 in Fig. 1(a) and 2(a). BT2’s ending scenario activating the SAS is the starting scenario of BT3. This means, BT2 and BT3 are interrelated, and so the root of BT3 can be merged with the leaf of BT2 to obtain an integrated model. The integrated BT for BT2 and BT3 has been shown in Fig. 4(b). Informally, integration is performed by merging the associated nodes. Similarly, performing node merging between all BTs (the rest are in Fig. 4(c-g)), we can obtain an integrated holistic model that represents the whole SAS ([3], [8]). This process reveals inconsistencies, incompleteness, ambiguities and redundancies in the requirements ([3], [9]).

![Mathematical Representation and Behavior Trees](image4.png)

3) Mathematical Representation of BTs: We use a very generic representation of Behavioral Model (BM), denoted by m, to refer to the mathematical representation of a BT [9]. As such, a BT node is referred to as Behavioral Unit (BU), denoted by u. A BU consists of a set of attributes \( \{a_k\} \) and so does an edge between two BUs. Each attribute \( a_k \) has a name, denoted by name\( (a_k) \), and a value, denoted by value\( (a_k) \). Fig. 4(a) shows a BM for BT3 having two BUs \( u_0 \) and \( u_1 \) connected by an edge \( e_{01} \).

4) Equivalence Relation between BUs: We defined a similarity measure between two BUs \( u \) and \( u' \) at [9]. The candidate BUs may not have the same set of attributes. Again, the values of the attributes may have compatibility issues. For example, state realisation and external/internal input/output...
are incompatible. Let $A$ denote the universal set of attributes; $A_u \subset A$ and $A_{u'} \subset A$ denote the set of attributes of the BUs $u$ and $u'$ respectively. Given that there exists no compatible attribute pair in $A_u$ and $A_{u'}$, the similarity measure $u \odot u' = (\sum_{i,j,k} (a_i \odot a_j) \cdot w_k) / \sum_k w_k$ for all $a_i \in A_u$, $a_j \in A_{u'}$ and $a_k \in A$ such that $\text{name}(a_i) = \text{name}(a_k) = \text{name}(a_j)$. Here, $a_i \odot a_j$ is the similarity of the attributes, and $w_k$ is the weight of $a_k$. For a given threshold $\alpha$, if $u \odot u' \geq \alpha$, then $u$ and $u'$ are equivalent. The weights and the threshold values may need to be configured locally for candidate BUs. For simplicity, let's assume a global configuration is used. Suppose the weights of $\text{component-name}$, $\text{behavior-name}$ and $\text{behavior-type}$ are 50%, 25% and 25% respectively; all the rest are 0%. Then the similarity between a BU $u_2$ of BT5 and the root BU $u_0$ of BT6 (Fig. 4(c, d)) becomes 0.75. If $\alpha = 0.75$ then the two BUs are equivalent. Note that the weights and the threshold values may be estimated by AI techniques using the data from different case studies (future work). However, the RA may choose these values according to their experience. From our experience, the weight of the $\text{component-name}$ should be higher than any other attributes for finding equivalence relations in BTs.

5) Integration Relation between BMs: A parent BM $m_p$ forms an integration relation with a child BM $m_c$, denoted by $R(m_p, m_c)$, if a BU $u_p$ in $m_p$ is equivalent to the root BU $u_0$ of $m_c$. Notice BT5 and BT6 in Fig. 4(c, d). They form an integration relation because $u_2$ of BT5 and $u_0$ of BT6 are equivalent as shown above. The root BU of a child BM may be equivalent to multiple BUs of its parent BM. Thus, they may form multiple relations. We use $R^k(m_p, m_c)$ to denote the $k$-th relation between $m_p$ and $m_c$. Depending on whether $u_p$ is the root, a branch or a leaf in the parent BM, we consider three types: root-root, branch-root and leaf-root relations [9]. Integration relations indicate a candidate merger of equivalent BUs and so a potential integration of two BMs into one.

III. INTERACTIVE REQUIREMENTS ENGINEERING

In a traditional RE process, natural language descriptions of a system’s requirements are first translated by the RA into a given model, in our case behavioral models (i.e. BMs). The RA then manually visits every model along with its interrelated models, and perform sanity and integrity checking. The RA might temporarly or finally make decisions about whether a particular BM can be included in the specification or not. Once a decision has been made for a BM, normally the RA then finds another BM, interrelated to the former, to repeat similar kinds of checking and decision making.

In this process mentioned above, the RA might face a number of difficulties. For example, depending on the scale of the system, the RA might not have an overall knowledge on the requirements. So the RA might go through all the BMs and relations each time to find a suitable BM for his next consideration. Moreover, the RA has to remember previous decisions made for other BMs. Sometimes, the process could go back and forth, affecting previous decisions in a cascading fashion. With the absence of a clearly defined mechanism and any aid from an automated system, these issues must be addressed by the RA through laborious, time-consuming and error-prone manual process. The whole process becomes even more complex for large-scale software intensive systems.

In this section, we provide an overview of a new framework of RE, referred to as Interactive Requirements Engineering (iRE). In iRE, the target integrated BM would be derived semi-automatically where the RA would get involved with the process in an interactive fashion. We do not opt for a fully automated system because explicit domain knowledge is required in various scenarios during the RE process.

The main concept of the iRE is to capture the meta-knowledge about the BMs and their integration relations into a semantic network model (SNM). The iRE system (Fig. 5) would provide an interactive interface to the RA who could utilise the SNM to make domain decisions. A defects detection and resolution framework (DDRF) would detect potential defects [9] using the SNM and the interrelated BMs, and would suggest the RA a set of potential resolutions. The DDRF could prioritise the defects, and advise what’s next to consider for making domain decisions. In the whole process, the RA may make changes in the BMs and/or their relations in the SNM. A change management framework (CMF) would synchronise the BMs and the SNM. The models integration framework (MIF) would automatically produce the integrated BM based on different configuration settings set by the RA.

![Fig. 5. Conceptual Model of iRE](image-url)

Fig. 6 shows the process flow and Fig. 7 shows the elaborated view of the iRE.

When the RA translates the informal requirements into a set of BMs, the iRE process would determine the integration relations between the BMs (previous work [9]). The iRE then would build the SNM which is a graph with the BMs as its vertices and the integration relations between BMs as its edges. In the SNM, the RA may want to choose the initialisation BM(s). Note an initialisation BM consists of behavioral scenarios to initialise a system. The SNM then would undergo a sanity checking, and decisions would be temporarily or finally made by the RA for its individual vertices i.e. the BMs to be ready for integration. Because of various types of relation in BMs, the temporary decisions could be revised back and forth, and...
thus could affect similar other decisions in a cascading fashion. In order to keep the cascading operation manageable, the SNM imposes a number of inference rules. Putting these altogether an SNM is called a normalised SNM when it passes the sanity checking and conforms to the inference rules. Expert opinion would be taken where necessary.

A normalised SNM is called a well-formed SNM when it is connected and it has one Init model. If the normalised SNM is not well-formed, the DDRF would detect potential defects; and would present the RA potential resolutions. The RA would make changes to the BMs and/or their interrelations if no decision has been made. Initially, all BMs are UN, (initialisation) BM by discretion. Suppose s/he selects m_2 as the Init BM. We say it is Included in the specification and the CMF updates the SNM by assigning IN state to m_2 (Fig. 8(b), double border: Init).

We also annotate each vertex by assigning a confidence of each BM to quantify the credence for inclusion in the specification. We assign confidence 1 if a BM is IN and 0 if EX. Otherwise, confidence of a BM is the confidence of its parent multiplied by the similarity measure of its corresponding integration relation. For simplicity, we consider the maximum value when multiple parents exist.

In addition to the similarity measure, an integration relation is annotated by its state. We consider two layers of thresholds: (1) IN threshold \( \alpha_{IN} \) and (2) UN threshold \( \alpha_{UN} \) where \( \alpha_{UN} < \alpha_{IN} \). We assume a relation is IN if similarity \( \geq \alpha_{IN} \), and EX if similarity \( < \alpha_{UN} \), and UN otherwise. For domain dependency, the decision is committed on RA’s consent.

### C. Develop Normalised SNM

After developing the SNM as shown in Fig. 8(b), the CMF derives a simplified version of the SNM where the relations are evaluated, minimised and organised, and the acceptability states and the confidences of the BMs are updated.

#### IV. ILLUSTRATION OF THE IRE FRAMEWORK

In this section, we illustrate the functionalities of the components of the iRE framework using the BMs of the SAS.

#### A. Determine Relations in the BMs

Let, \( m_1 \) to \( m_7 \) denote the BMs for R1 to R7 respectively. Using the same similarity measuring formula and parameters as in Section II, Table II shows the relations in them. Column BMs shows parent and BMs (parent and child) having an integration relation, column Relations indicates the relation type, column \( u_p \) shows the parent integration BU (note that the child integration BU \( u_c \) is omitted since it is always the root BU of the child BM), and column \( u_p \oplus u_c \) shows the similarity measure between \( u_p \) and \( u_c \).

#### B. Develop SNM from the interrelated BMs

The SNM consists of vertices that represent the BMs and the edges that represent the integration relations between BMs. Fig. 8(a) shows a graph where the vertices refer to the BMs, and the edges refer to their relations annotated with similarity. We gradually develop the SNM as shown in Fig. 8(b).

Firstly, we annotate each vertex by assigning an acceptability state. We consider three states: Included (IN), Excluded (EX) and Undecided(UN). A BM is IN if the RA accepts it as valid, and thus the BM is included in the specification. A BM is EX if the RA does not accept it as valid and so is excluded from the specification, and it is UN if no decision has been made. Initially, all BMs are EX. To start, s/he selects a BM as an Init (initialisation) BM by discretion. Suppose s/he selects \( m_2 \) as the Init BM. We say it is Included in the specification and the CMF updates the SNM by assigning IN state to \( m_2 \) (Fig. 8(b), double border: Init).
Fig. 9(a) shows the root-root relations between \( m_3 \) and \( m_4 \). In this case, the integrated BM would be the same irrespective of which relation we consider. So, a root-root relation can safely be eliminated. From the candidate BMs, we choose a BM, referred to as the leader model, that is either an Init model or the closest model among others from an Init model. We then eliminate the relation incoming to that leader model. However, the elimination is committed upon the RA’s consent. If there is a tie, the model having more (or equal) children is chosen as the leader model. In Fig. 8(b), \( m_4 \) is the leader among \( \{m_3, m_4\} \). Even if their distance from the Init model (\( m_2 \)) is 1, \( m_4 \) has more children (2) than \( m_3 \) (1). Similarly, \( m_6 \) is the leader among \( \{m_5, m_7\} \). Therefore, the relations between \( m_3 \) & \( m_4 \), and \( m_7 \) & \( m_6 \) are eliminated as shown in Fig. 9(b).

A BM can be validated before assigning an acceptability state. For example, the scenarios for the Init model \( m_2 \) can be examined to extract a set of rules:

1) (L1) if \{\}, then ¬SAS-Activated. That is, by default SAS is not Activated.
2) (L2) if SET-BUTTON-Pressed, then SAS-Activated. This rule overrides rule L1.

The above rules have been expressed in Clausal Defeasible Logic (CDL) [13], which is implementable [14]. We can form a rule-set, from the above L1 and L2 rules i.e. rule-set = \{L1, L2\}. Since incremental refinement of a (conflicting) rule set can be performed in non-monotonic logic ([15], [16]), the rule-set can be gradually developed from the BMs. In this paper, we assume that the BMs are valid against the rule-set.

Suppose the relation between \( m_5 \) & \( m_6 \) is accepted by the RA. As shown in Fig. 11(b), we can then infer IN states for \( m_3, m_4, m_5 \) and \( m_6 \). We further infer UN state for \( m_7 \) since neither its parent nor the relation is EX. Lastly, we infer EX for \( m_1 \) since it is no way connected with the Init model.

Further, Fig. 11(a) shows a circular relation consisting of \( m_2, m_4, m_5 \) and \( m_6 \). Now, if these BMs are integrated, it over-specifies the scenarios of the behaviors. Fig. 12 (a) shows the integrated model where the border lines show the same sequence of behaviors \((u_0, u_1, u_2, u_3)\) and \((u_10, u_{11}, u_{12}, u_{13})\). To avoid this over-specification, a relation must be eliminated. We again select \( m_2 \) as the leader model. Fig. 12(b) shows the updated SNM, which we refer to as normalised SNM.

D. Defects Detection and Resolution

The SNM aids finding out defects in the requirements earlier. The BM \( m_3 \) gets EX state and it does not have any relation, so requirement R1 is incomplete. Please refer to our
previous paper [9] for techniques of potential defects detection. Defects may be resolved based on a set of resolution heuristics and suggestions could be presented to RA for making domain decisions, which are kept as future work.

E. Develop Well-formed SNM

This property determines whether the system requirements can be integrated into one formal specification or not. A normalised SNM is well-formed if (1) the number of Init models in the SNM is 1, (2) all models have IN states, and (3) all existing relations have IN states.

The RA may want to analyse why m₁ could not form a relation. The BM m₁ is about sending the trip signal when a motion is detected. It is easily understood that a precondition is missing. Suppose RA thinks that m₁ should happen when the SAS is activated (Fig. 13). Now, it forms root-root relations (similarity 1) with m₃ and m₄, and leaf-root relation with m₂ (Fig. 14(a)). Fig. 14(b) shows the normalised SNM.

If the RA accepts the relation between m₅ and m₇, the BM m₇ becomes IN. Fig. 15 shows the updated SNM, which is well-formed as well.

F. Derive Integrated Model

The well-formed SNM can be traversed from the Init model utilising breadth-first-search, so an integrated BM can be derived as well. Fig. 16 shows the integrated BM of the SAS. The integration of BMs has been formalised at [10].
UML diagram such as Class Diagram, Sequence Diagram and State-chart Diagram. However, UML diagrams [22] represent a variety of aspects of the system across several diagrams. Further, some UML diagrams do not have formal semantics. As these diagrams have different syntax and semantics, it often becomes difficult to define relationships between the diagrams. In UML, a number of overlapping partial views between diagrams makes it hard to formalise the impact of a change in the requirements in the modeling stage [23].

Göknil et al. [24] formally defined different types of relation in the requirements using first-order logic and developed change impact rules based on a requirement metamodel. However, they did not show how the requirement metamodel can be utilised for the management of the requirements models at earlier stage of system modeling. Wen and Dromey ([25]) proposed a traceability model to determine the impact of a change on the architecture and design of the system by embedding the changes in the specification behavior tree. To the best of our knowledge, there exists no framework that utilises a semantic network of behavioral models as knowledge-base in the earlier stage of requirements modeling.

Our proposed mechanism is highly inspired by the techniques applied to an epistemology-based system such as Ellis’s belief system [26] from the Philosophy field and Doyle’s Truth Maintenance System (TMS) [27] from the AI field. Ellis ascribes a sentence $A$ of a language $L$ as true $T$, false $F$ or an absence of firm belief $X$ and uses a number of acceptability criteria to determine the equilibrium (e.g. consistency, normalisation) condition. Doyle used a semantic network model as the epistemic state of belief, where the nodes refer to a set of objects of beliefs and links refer to the relations of the beliefs. Doyle also developed truth maintenance mechanisms to maintain the beliefs and the reasons of beliefs. In our model, justification of the acceptability of a BM depends on the validity against the rule-set and the related BMs. Our developed SNM also captures the confidence level of each of the BMs. Three types of states, along with the confidence level, explicitly portray a mental judgement on a BM.

**VII. CONCLUSIONS AND FUTURE WORK**

In this paper, we presented an overview of a new approach of requirements engineering. We used a semantic network model as a knowledge-base to capture the meta-level information of the system requirements and their relationships. We then illustrated how the SNM can support the requirements engineering in a seamless fashion. In near future, we would like to formalise each of the components (SNM, CMF, MIF and DDRF) of the iRE framework along with their implementation. We also plan to perform experimental evaluation of the iRE framework by emulating large-scale software intensive systems. Our plan also includes evaluating the approach using large-scale real case studies collaborating with industries.

**REFERENCES**


