Formalisation of the Integration of Behavior Trees

Kushal Ahmed, M. A. Hakim Newton, Lian Wen, Abdul Sattar
Institute for Integrated and Intelligent Systems
Griffith University, Nathan, QLD 4111, Australia
{k.ahmed, hakim.newton, l.wen, a.sattar}@griffith.edu.au

ABSTRACT
In this paper, we present a formal definition of the integration of the requirements modeling language Behavior Trees (BTs). We first provide the semantic integration of two interrelated BTs using an extended version of Communicating Sequential Processes. We then use a Semantic Network Model to capture a set of interrelated BTs, and develop an algorithm to integrate them all into one BT. This formalisation facilitates developing (semi-)automated tools for modeling the requirements of large-scale software intensive systems.

Categories and Subject Descriptors
D.2.1 [Software Engineering]: Requirements/Specifications—requirements modeling, process automation

Keywords
Behavior Trees; Behavior Engineering; Integration; Merge; CSP; Semantic Network Model

1. INTRODUCTION
Requirements Engineering (RE) often starts with a set of scattered and unstructured system requirements with a view to achieving an integrated and structured formal specification. Unsurprisingly, such requirements may be filled with problems, such as ambiguity, inconsistency, redundancy, and incompleteness. As a result, the process of transforming the requirements into a formal specification is extremely complicated. The transformation process must therefore be able to identify issues in the requirements in a way easy to understand, and the model must be structured such that it can be cross referenced with the original requirements document.

Behavior Engineering (BE) was developed to address this problem [5,15]. It uses Behavior Tree (BT) as the graphical notation. A BT contains a range of constructs that cover state-based manipulations, as well as more abstract concepts such as synchronisation and message passing, along with typical concurrency, choice and iteration control structures familiar to specification and programming languages.

Each requirement is translated into one or more BTs, and each node in the tree is tagged with the requirements identifier allowing traceability back to the original informal requirements. BTs may have syntactically matching constructs. Therefore, they may then be progressively integrated into one holistic model of the system, which serves as a formal specification. This process may reveal inconsistencies, redundancies, incompleteness, and ambiguities. BTs have been adopted for industrial use, in particular Raytheon Australia [1], who invested resources to developing a BT editor [10]. There are many other tools on BTs [14,18,20,23], some of them support validation and verification as well.

However, the integration of the BTs is still informal, ad-hoc and manual. Colvin and Hayes [4] provided the semantics of a BT using Hoare’s process algebra Communicating Sequential Processes (CSP) [9,19] as its base, which is a well established and elegant formal notation for describing interactions between concurrent processes. By extending it, they defined CSP∗ to include state-based constructs such as tests and updates, and message passing facility similar to publish/subscribe models of the communication [6]. In this paper, we first provide the semantic integration of two interrelated BTs using an extended version of CSP∗. We then use a Semantic Network Model (SNM) [3] to capture a set of interrelated BTs, and develop algorithm to integrate them all into one BT. This formalisation facilitates developing (semi-)automated tools for modeling the requirements of large-scale software intensive systems.

Paper organisation: Section 2 briefly describes BTs and their interrelations, CSP∗ and SNM; Section 3 formalises the integration of two interrelated BTs; Section 4 extends the formalism for a set of BTs; Section 5 explores related literature; and finally Section 6 presents our conclusion.

2. BACKGROUND
In this section, we briefly describe the constructs of BTs and their interrelations, CSP∗ and SNM.

BTs and their interrelations: Fig. 1(a) shows a BT of a requirement (ID: R2) of a Security Alarm System (SAS) [3,17]. It mentions the initialisation scenarios for activating the SAS. Fig. 1(b) displays the contents of a BT node. Each node is associated with a component having a behavior. The type of behavior may be internal input (≫), external input (≫≪), internal output (≪≫), external output (≪≪), event (????), selection, (??), or state realisation ([][]) etc.
Figure 1: Behavior Tree and Attributes of a node

The node operators such as reversal ((\)), branch kill (--) and synchronisation (=) are used to indicate control flow beyond simple sequential execution and branching. The nodes can be connected in different ways such as sequential, parallel, atomic and alternative ([4,5]).

Notice BT2 in Fig. 1 (a) and BT3 (ID: R3) in Fig. 2(a). BT2’s ending scenario activating the SAS is the starting scenario of BT3. This means, BT2 and BT3 are interrelated, and can be integrated into one BT (Fig. 3(a)).

3. FORMAL INTEGRATION

In this section, we first provide the semantics of a BM. We then formalise the integration of two interrelated BMs.

3.1 Semantics of a BM: Extension of CSP

A BM consists of BUs connected by edges. The attributes of each of these entities may have operational semantics. For example, the behavior-type attribute of a BU indicates the type of action such as state realisation, selection, guard, input or output for the component denoted by component-name. The node-operator attribute indicates recursion, interrupt, restart or synchronisation.

For example, the behavior-type attribute of a BU indicates atomic, sequential, parallel or alternative step from one action to another. An edge may have timing constraints. Thus, an edge represents the characteristics of the step between actions such as actions performed non-interleaving (atomic), sequential, parallel or external choice (alternative), which may further be executed after a fixed amount of time delay. CSP\textsubscript{\sigma} [4] does not consider the edge as an entity, however it provides semantics of atomic, sequential, parallel and alternative steps.

Fig. 4 shows a sample BM having BUs connected with edges. CSP\textsubscript{\sigma} defines semantics of the BM considering the attributes (e.g. behavior-type) of the BUs and the type of edges. It also defines non-sequential execution of the processes for node-operator attribute. On top of it, we can extend the CSP\textsubscript{\sigma} as follows.

Let $P_u$ denotes the CSP\textsubscript{\sigma} process starting from the CSP\textsubscript{\sigma} action characterised by the BU $u$. As such, $P_u$ denotes the CSP\textsubscript{\sigma} of the BM shown in Fig. 4. Here, the root BU $u_0$ has three children $u_1$, $u_2$ and $u_3$ connected by $e_{0,1}$, $e_{0,2}$ and $e_{0,3}$ respectively. If the CSP\textsubscript{\sigma} action for $u_0$ is performed, the CSP\textsubscript{\sigma} process $P_{u_0}$ may behave as $P_{u_1}$, $P_{u_2}$ and/or $P_{u_3}$ after following the steps characterised by the edges $e_{0,1}$, $e_{0,2}$ and $e_{0,3}$ respectively. Therefore, $P_{u_0}$ can be written by the following way:

$P_{u_0} ::= \llangle u_0 \mid R(u_0, e_{0,1}) \rightarrow P_{u_1} \mid R(u_0, e_{0,2}) \rightarrow P_{u_2} \mid R(u_0, e_{0,3}) \rightarrow P_{u_3} \rrangle$ s.t. $u_j \in \text{childs}(u_0)$

(1)

Symbol $\llangle u_j \mid R(u_0, e_{0,j}) \rightarrow P_{u_j} \rrangle$ indicates concurrent processes for each $u_j$.

DEFINITION 1 (SEMANTICS OF A BM). The semantics of a BM is a CSP\textsubscript{\sigma} with BUs as the CSP\textsubscript{\sigma} actions and edges as the steps from one action to another. Let $P_u$ denotes the CSP\textsubscript{\sigma} process starting from the action characterised by the BU $u$, which has children BUs denoted by $\text{childs}(u)$, then, $P_u ::= \llangle u \mid R(u, e_{u,j}) \rightarrow P_{u_j} \rrangle$ where $u_j \in \text{childs}(u)$

(2)
3.2 Integrating BMs: Merging CSP processes

If two BMs have root-root, branch-root or leaf-root relation, they can be integrated into one BM by merging their equivalent BUs pair into one BU. By Definition 1, a BM represents a CSP process and each BU represents a CSP action. As a result, integrating two BMs means merging two CSP processes. This further implies that the CSP actions represented by the equivalent BUs can be merged into one CSP action. In that case, the attributes that characterize the merged CSP action would depend on the merging of the attributes of the equivalent BUs. We refer to the merged CSP action as the least common action (lca).

Definition 2 (Least Common Action). If two BMs have an integration relation, the CSP actions represented by the equivalent BUs, and can be merged into one CSP action, referred to as the least common action (lca) of the two CSP processes represented by the BUs respectively.

In addition, the equivalent BUs may have child BUs which themselves may again be equivalent and so on. That means, the CSP processes for the child equivalent BUs can again be merged into one CSP process (Procedure 4); which may continue till a leaf BU is reached. Thus a refinement of CSP processes must follow after the lca to achieve a merged CSP process that is more defined than the original two CSP processes. In this section, we first illustrate the integration and then formalise it using the extended version of CSP processes.

3.2.1 Illustration of the Integration

The BU of BT5 (Fig. 3(b)) is equivalent to the root BU of BT6 (Fig. 3(c)). Therefore, they form an integration relation (Fig. 5(a)). So they can be integrated into one BT. The integration starts from integrating the BUs and (Fig. 5(b)). Note that the behavior type of these two BUs are different: state realisation and event. Here, state realisation has high precedence over event. So the integrated BU takes state realisation value of the behavior type attribute. After integrating these BUs, the child BUs of BT5 and of BT6 of these former BUs are equivalent again (Fig. 5(b)).

Therefore, the edges of BT5 and of BT6 are integrated first and then the BUs of BT5 and of BT6 are integrated. Now, the child BUs of BT5 and of BT6 are further equivalent (Fig. 5(c)). The integration ends after integrating the edges of BT5 and of BT6, and the BUs of BT5 and of BT6 (Fig. 5(d)). Thus, the integration starts by integrating the equivalent BUs which causes the integration relation to be formed between the BMs. Then, the integrated BM goes through further refinement by integrating the equivalent child BUs and so on.

3.2.2 Integration for root-root relation

Let, and represent the root BUs of the BMs having root-root relation. From Definition 1,

\[ P_{u_2} := (u_2, e_{i,j'}) \rightarrow P_{u_i} \] where \( u_i \in \text{childs}(u_2) \) (3)

\[ P_{u_1} := (u_1, e_{i,j'}) \rightarrow P_{u_j} \] where \( u_j \in \text{childs}(u_1) \) (4)

Let, \( u_i = u_p, u_j = u_q \) as shown in Fig. 6. Suppose \( u_i \cup u_j \) (= \( u_{ij} \)) denotes their integration. Assume the child BUs \( u_i \) and \( u_j \) are again equivalent. Therefore, we have to integrate the edges \( (e_{i,j'} \oplus e_{j',i'}) = e_{i,j'} \), and the BUs \( (u_i \cup u_j \rightarrow u_{ij}) \) as indicated by the dotted boxes in Fig. 6. Now,

\[ P_{uj} := (u_{ij}, e_{i,j'}) \rightarrow P_{ui} \] where \( u_k \in \text{childs}(u_{ij}) \) (5)

The procedure starts from integrating the BUs and and further refines the integrated BM. Therefore, the attributes that characterize the merged CSP action would depend on the merging of the attributes of the equivalent BUs. We refer to the merged CSP action as the least common action (lca).
3.2.3 Integration for branch-root relation

Let, \( u_p \) and \( u_c \) represent the equivalent BUs of the BMs having branch-root relation. Suppose, \( u_c \) denotes the root of the parent BM. Let, we rewrite the CSP\(_{P} \) process \( P_u \) as \( P_u := P_{u_p} \sim P_{u_c} \). Let, \( u_1 = u_p, u_j = u_c \) as shown in Fig. 7. The equivalent BUs and corresponding edges have to be integrated as indicated by the dotted boxes in Fig. 7. \( P_{u_1} \) of \( P_u \) and \( P_{u_j} \) can be formed similarly by Equation 7.

\[
P_{u_1} := \| u_k \| ((u_p \oplus u_c, e_{p,c,k}) \rightarrow P_{u_k})
\]

where \( u_k \in \{ \text{children}(u_p) \cup \text{children}(u_c) \} \)

Figure 7: Sample BMs having branch-root relations

**Procedure 2** (Integrating branch-root related BUs).

1. Let two BMs have branch-root relation, the CSP\(_{P} \) processes \( P_{u_p} \) and \( P_{u_c} \) represented by the BUs can be merged into one CSP\(_{P} \) process, denoted by \( P_u := P_{u_p} \oplus P_{u_c} \), in two phases:
   1. Least Common Action: The merging begins with producing the lca of the two processes \( P_{u_p} \) and \( P_{u_c} \). Thus,
      \[
P_{u_{pc}} := \| u_k \| ((u_p \oplus u_c, e_{p,c,k}) \rightarrow P_{u_k})
      \]
      where \( u_k \in \{ \text{children}(u_p) \cup \text{children}(u_c) \} \) (10)
   2. Refinement: Let, \( u_i = u_p, u_j = u_c, \) and so \( u_{ij} = u_{pc} \). For each equivalent BUs pairs \( u_i \) and \( u_j \) such that \( u_i \in \text{children}(u_i) \) and \( u_j \in \text{children}(u_j) \), the refinement steps are:
      (a) Let, we rewrite \( P_{u_i} \) as \( P_{u_c} := P_{u_i} \sim P_{u_{ij}} \). Here,
          \[
P_{u_{ij}} := \| u_k \| ((u_i \oplus u_j, e_{i,j,i}, e_{i,j,k}, e_{ij,k}) \rightarrow P_{u_k}, e_{i,j,k} \rightarrow P_{u_k}) \]
          where \( u_k \in \{ \text{children}(u_i) \cup \text{children}(u_j) \} \) (11)
      and \( u_k \in \{ \text{children}(u_i) \cup \text{children}(u_j) \} \) - \{ \( u_i \), \( u_j \) \}
   (b) Assign \( u_i = u_i' \) and \( u_j = u_j' \), and perform Refinement.

3.2.4 Integration for leaf-root relation

Let, \( u_p \) and \( u_c \) represent the equivalent BUs of the BMs having leaf-root relation. Suppose, \( u_c \) denotes the root of the parent BM. Since it is a leaf-root relation, there exists no child of the leaf BM. Let, we rewrite the CSP\(_{P} \) process \( P_u \) as \( P_u := P_{u_p} \sim P_{u_c} \). Fig. 8 shows the merging is performed by integrating the equivalent BUs \( u_p \) and \( u_c \).

\[
P_{u_{pc}} := \| u_k \| ((u_p \oplus u_c, e_{p,c,k}) \rightarrow P_{u_k}) \]

Figure 8: Sample BMs having leaf-root relations

**Procedure 3** (Integrating leaf-root related BUs).

If two BMs have leaf-root relation, the corresponding processes \( P_u \) and \( P_{u_c} \) can be merged into one CSP\(_{P} \) process, denoted by \( P_u := P_{u_p} \oplus P_{u_c} \), by producing the lca:

3.3 Integrating BUs: Merging CSP. Actions

If two BUs are equivalent, the integration of them represents a merged CSP\(_{P} \) action. It is characterised by the attributes of the integrated BU, which we get by integrating the attributes of the given BUs.

**Procedure 4** (Integrating BUs: \( u_p \oplus u_c \)). If the BUs \( u_p \) and \( u_c \) are equivalent, the corresponding CSP\(_{P} \) actions can be merged into one CSP\(_{P} \) action that is characterised by the attributes of the BU \( u_{pc} := u_p \oplus u_c \) formed by integrating the attributes of \( u_p \) and \( u_c \) by the following rules:

1) If there exists a pair of attributes \( a_i \in u_p \) and \( a_j \in u_c \) such that \( \text{name}(a_i) = \text{name}(a_j) \), then an attribute \( a_k \in u_{pc} \) is formed such that \( a_k = a_i \oplus a_j \).
2) If there exists an attribute \( a_i \in u_p \) such that \( \text{name}(a_i) \neq \text{name}(a_k) \) for all attribute \( a_k \in u_c \), then an attribute \( a_k \in u_{pc} \) is formed such that \( a_k = a_i \).
3) If there exists an attribute \( a_i \in u_c \) such that \( \text{name}(a_i) \neq \text{name}(a_k) \) for all attribute \( a_k \in u_p \), then an attribute \( a_k \in u_{pc} \) is formed such that \( a_k = a_i \).

Due to similarity we omit the formalisation of integrating edges i.e. merging the steps between the CSP\(_{P} \) actions.

3.4 Integrating attributes

When integrating two BUs or edges, we essentially integrate the attributes of these entities. If the names of the attributes \( a_i \) and \( a_j \) are same, we form an integrated attribute \( a_k = a_i \oplus a_j \). The value of the integrated attribute \( a_k \) depends on the values of the attributes \( a_i \) and \( a_j \). If the values are same, then the integrated value is also the same. Otherwise, the value of the integrated attribute is formed by many different operations between the original values.

**Procedure 5** (Integrating attributes: \( a_i \oplus a_j \)). If two attributes \( a_i \) and \( a_j \), given that \( \text{name}(a_i) = \text{name}(a_j) \), are integrated into one attribute \( a_k \), where \( \text{name}(a_k) = \text{name}(a_i) \) and \( \text{name}(a_k) = \text{name}(a_j) \), then \( \text{value}(a_k) \) is formed by the following rules:

1) If \( \text{value}(a_i) = \text{value}(a_j) \), then \( \text{value}(a_k) = \text{value}(a_i) \).
2) Otherwise, \( \text{value}(a_k) = \text{value}(a_i) \oplus \text{value}(a_j) \).

The integration of the values i.e. \( \text{value}(a_i) \oplus \text{value}(a_j) \) can be performed in many different ways.

**Choice:** The value of the integrated attribute may be selected from the values of the original values. For example, the values of the behavior type attribute may be different.

**Combination:** The value of the integrated attribute may be the combination of the original values. For example, the value of traceability link attribute may be different. The values may be R2 and R3. In this case, the value of the integrated attribute would be the combination of those values i.e. \( \{R5, R6\} \). We say that the property of the attribute traceability link is combination.
Mapping: The values of the attributes may be syntactically different, but semantically same. For example, the component name of a component may have aliases, but all these aliases refer to the same component.

Thus, the integration of the values may be the mapped value from the original values, intersection of the values, mathematical operation and so on. Which strategy should be taken depends on the domain information.

3.5 Theorems and Properties

**Corollary 1** (Closure of Merging). (1) The merging of the two CSPs actions by Procedure 4 produces another CSP action and (2) the integration of two attributes by Procedure 5 produces another attribute.

**Theorem 1** (Preservation of Semantics). The semantics of the integrated BM by Procedures 1, 2 and 3 is consistent with the semantics of the given two input BMs.

**Theorem 2** (Polynomial Complexity). Given two BMs having edge sets $E_1$ and $E_2$, the worst-case complexity of (1) Procedure 1 is $O(\min(e_1, e_2))$, (2) Procedure 2 is $O(e_2)$ and (3) Procedure 3 is $O(1), e_1 = |E_1|, e_2 = |E_2|$.

**Theorem 3** (Closure of Merging CSP$\sigma$ processes). The merging of the two CSP$\sigma$ processes by Procedures 1, 2 and 3 also produce a CSP$\sigma$ process.

**Theorem 4** (Optimal Refinement). Procedures 1, 2 and 3 produce an optimally refined CSP$\sigma$ process from the given two CSP$\sigma$ processes.

Properties of the merging of CSP$\sigma$ processes: We get the following properties:

1) Reflexive: $P_u \oplus P_u = P_u$. 3) Partial Symmetry: $P_u \oplus P_v = P_v \oplus P_u$, if the corresponding BMs have root-root relation.

Properties of the merging of CSP$\sigma$ actions, edges and attributes: Each operation shares the same set of properties. We mention the properties for CSP$\sigma$ actions:

1) Reflexive: $u_i \oplus u_i = u_i$. 2) Symmetric: $u_i \oplus u_j = u_j \oplus u_i$. 3) Associative: $u_i \oplus u_j \oplus u_k = (u_i \oplus u_j) \oplus u_k = u_i \oplus (u_j \oplus u_k)$. Proofs will be available in the longer version of the paper.

4. INTEGRATION OF A SET OF BMS

In this section, we use an SNM of the interrelated BMS and develop an algorithm to integrate all the BMS into one BM. Let, $m_1$ to $m_7$ denote the BMS of the SAS. Suppose they are interrelated to each other as shown in Table 1. The column BMS shows the pair of parent and child BMS. Relations($u_p$, $\circ$) shows the type of relation, parent BU ($u_p$) and the similarity measure ($\circ$) of the parent and child BUs.

Table 1: Relations of the BMS of the SAS.

<table>
<thead>
<tr>
<th>BMSs</th>
<th>Relations($u_p$, $\circ$)</th>
<th>BMSs</th>
<th>Relations($u_p$, $\circ$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_2$, $m_3$</td>
<td>leaf-root($u_{a}$, 1)</td>
<td>$m_2$, $m_4$</td>
<td>leaf-root($u_{a}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_4$</td>
<td>root-root($u_{o}$, 1)</td>
<td>$m_4$, $m_3$</td>
<td>root-root($u_{o}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_5$</td>
<td>leaf-root($u_{a}$, 1)</td>
<td>$m_5$, $m_2$</td>
<td>leaf-root($u_{a}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_6$</td>
<td>branch-root($u_{b}$, 1)</td>
<td>$m_5$, $m_7$</td>
<td>branch-root($u_{b}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_7$</td>
<td>root-root($u_{o}$, 1)</td>
<td>$m_7$, $m_6$</td>
<td>root-root($u_{o}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_8$</td>
<td>root-root($u_{o}$, 1)</td>
<td>$m_1$, $m_3$</td>
<td>root-root($u_{o}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_9$</td>
<td>root-root($u_{o}$, 1)</td>
<td>$m_3$, $m_1$</td>
<td>root-root($u_{o}$, 1)</td>
</tr>
<tr>
<td>$m_2$, $m_{10}$</td>
<td>leaf-root($u_{a}$, 1)</td>
<td>$m_5$, $m_2$</td>
<td>branch-root($u_{b}$, 1)</td>
</tr>
</tbody>
</table>

Fig. 9 shows a sample SNM created from the Table 1. We assumed that $m_2$ is the initialisation (Init) model. Initially, $m_2$ is IN, and all other BMSs are EX. For a BM, 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.

Further, a relation is annotated by its acceptability state. We consider two layers of thresholds namely (1) IN threshold $\alpha_N$ and (2) UN threshold $\alpha_M$ where $\alpha_M < \alpha_N$. We assume a relation is IN if similarity $\geq \alpha_N$, and EX if similarity $< \alpha_M$, and UN otherwise. Assume that $\alpha_N = 1$ and $\alpha_M = 0.75$.

**Theorem 6** (Soundness & Completeness). (1) Given a well-formed SNM, Procedure 6 produces an integrated BM in polynomial time, (2) Procedure 6 can be applied to any well-formed SNM to produce an integrated BM.
5. RELATED WORK
We classify the related work into two categories: Formalisation of BE, and Integrating Software Models.

Formalisation of BE: Colvin and Hayes [4] provided the semantics of the BT using CSP.* We extended the semantics by including the edge between nodes as an entity in CSP.* Winter et al. [24] developed a framework of rules for integrating requirements using a notation-independent graphical model. Though they discussed overall broad approach of BTs integration, the rules do not relate the integration with BT’s semantics. They also did not develop the techniques to automatically integrate a set of BTs.

BT has been used for model-checking to assess the safety requirements [12], analyse role-based access control [26], perform Failure Modes and Effects Analysis (FMEA) [8], to identify the combinations of component failures by Cut Set Analysis (CSA) [11], verify large systems using a slicing techniques [25] and so on. Myers et al. [16] simulated behavioral scenarios using BTs into Modelica framework.

Integrating Software Models: In earlier stage (9), [13], the main focus was to do parallel composition of models that explains how two different components work together. Since different UML diagrams have different syntax and semantics, merging of those models are mostly manual ([7], [22]). Uchitel and Chechik [21] worked on merging two partial behavioral models of the same component to produce an elaborated version of both original models. Our works differ from these works in composing two behavioral models depicting state-based and scenario-based behavioral scenarios of more than one interacting components of the software system, which may have overlapping system behaviors, to obtain a more compact version of the original two.

6. CONCLUSION
In this paper, we provided a formal definition of the integration of the BTs. We first formalised the semantic integration of two interrelated BTs using an extended version of CSP.* We then formalised the integration of a set of BTs. Our future research focus is to formalise specification process where we would like to semi-automate the tasks to convert an integrated BT into a specification BT which can be validated by simulation and verified by model-checking.

7. REFERENCES