Generating Safety Verification Conditions Through Fault Tree Analysis and Rigorous Reasoning

Jeffrey Joyce; Raytheon Systems Canada Ltd.; Richmond, BC, Canada

Ken Wong; Department of Computer Science, University of British Columbia; Vancouver, BC, Canada

Keywords: safety requirements, system safety engineering.

Abstract

An approach based on informal, rigorous reasoning is described as a means of discovering "safety verification conditions" (SVCs). The approach can be carried out at various levels of detail. When the approach is carried out at the level of a "black box" view of the system, the result is a set of system safety requirements. The approach has similarities with Fault Tree Analysis (FTA) and Software Fault Tree Analysis (SFTA). Like FTA, a given hazard is traced backwards through the system to cover all the ways in which a hazardous condition can occur. Like SFTA, a "proof-by-contradiction" style reasoning is employed. The approach is illustrated by a detailed (hypothetical) chemical factory information system which is similar to other safety-related real-time information systems.

Introduction

This paper describes the use of semi-formal mathematical techniques for the stepwise refinement of hazard definitions into "safety verification conditions" (SVCs). Carrying out the derivation process down to the level of a "black box" view of the system results in SVCs which can be regarded as system safety requirements. The development of safety requirements is mandated in most system safety standards, such as MIL-STD-882C (ref. 1) and IEC 1508 (ref. 2).

The semi-formal approach described in this paper is intended to improve upon the accuracy and thoroughness of an *ad hoc* approach that relies entirely on engineering judgment. Our approach is particularly useful for deriving SVCs for a real-time system. Time-dependent hazards will typically involve subtle relationships between various temporal constraints. An important aspect of our approach is the systematic derivation of key temporal constraints and relationships from hazard definitions. This improves upon an *ad hoc* approach where these temporal relationships may be especially elusive.

Our approach is similar to Fault Tree Analysis (FTA) (ref. 3), as well as to Software Fault Tree Analysis (SFTA) (ref. 4). Our approach is similar to FTA in the sense that it begins with an assumption that the hazardous condition has occurred. From this point, the analyst uses FTA to work "backwards" to systematically cover all the possible ways in which this condition might have arisen. Like Software Fault Tree Analysis (SFTA), our approach uses a style of reasoning known as "proof-by-contradiction" to show that each disjunctive branch of the tree leads to a logical contradiction. However, the purpose of our approach, unlike FTA or SFTA, is to derive SVCs for the system and system components.

The approach described in this paper is one of the results of a university-industrial collaborative project called "formalWARE" (ref. 7). The approach is illustrated in this paper by means of a hypothetical chemical factory information system. This example has similarities with other safety-critical information systems such as Air Traffic Management (ATM) systems (ref. 5). In the development of the chemical factory example, the use of rigorous, mathematical reasoning led to the discovery of several system-level SVCs, including some conditions that we believe are not obvious.

The approach outlined in this paper provides a systematic approach to the discovery of system-level SVCs. The following section introduces the example used in this paper to illustrate our approach. Next, the inadequacies of an *ad hoc* approach to the derivation of SVCs is presented. This is followed by a sketch of the stepwise process used to generate SVCs for a particular hazard identified for this system. A discussion of some related work is presented, followed by a summary of the paper.

Example - Chemical Factory Information System

For illustrative purposes, this paper focuses on a hypothetical information system for a chemical factory. The chemical factory information system is similar to other real-time information systems, like ATM systems, in that environmental data is received, processed and displayed to operators. The operators then make safety-critical decisions based on the information displayed by the system.

System Description: The factory consists of a set of reactor vessels equipped with sensors that record data such as temperature and pressure. The sensors are connected through a LAN to the chemical factory information system, which runs on a central server and a set of workstations. The information system maintains and processes the vessel information it receives over the LAN and displays it on the workstation monitors.

Safety-Related Hazard: Following a process such as the system safety engineering process outlined in Reference 6, we assume that the display of an "invalid" value for the temperature of a vessel is identified as a system hazard for the chemical factory control and monitoring system.

> **Hazard:**
> An invalid temperature, D, is displayed for vessel V at time T.

It may be assumed that the identification of this hazard resulted from some earlier analysis, which shows that the display of an invalid value for the temperature of a vessel, in combination with other conditions, could lead to a mishap such as a fire or explosion.

Hazard Analysis: Based on analysis of the hazard, it may be assumed that the simple failure to display a valid vessel temperature will not cause the hazard. When the system is unable to display a valid temperature for a particular vessel, the system is required to mark the temperature field for this vessel as "unavailable". Even though the appearance of "unavailable" on the operator display in the temperature field for some particular vessel may be a result of a system fault, it has been determined (for this hypothetical example) that its appearance is not unsafe. This determination may, for instance, be partially based on the assumption that the human operator should not be misled by the "unavailable" indication in the same way that he or she may be misled by an invalid value. Hence, the term "invalid" is used in the definition of this particular hazard to refer to a temperature that could be mistaken as a valid temperature of the vessel. In particular, the appearance of "unavailable" on the operator display in the temperature field for some particular vessel is excluded from the definition of this hazard.

Further analysis of the hazard provides a more precise definition of the sense in which a displayed temperature value may be "invalid":

> The value displayed at time t as the temperature of a particular vessel is "invalid" if and only if the value displayed at time t for the vessel has not been within MAX_DISP_TEMP_DIFF degrees of the actual temperature of the vessel at some time t' within MAX_DISP_TEMP_STALE milli-seconds before time t.

where MAX_DISP_TEMP_DIFF and MAX_DISP_TEMP_STALE are "requirements-level" system constants defined as follows:

- MAX_DISP_TEMP_DIFF is the tolerance allowed for displayed temperatures, i.e., the maximum difference allowed between the actual temperature of the vessel and the value displayed to the operator;
- MAX_DISP_TEMP_STALE is the maximum amount of time that a value may be displayed before it is considered "stale".

Hence, a displayed value may be invalid because it has been corrupted or because it has become stale.

System-Level SVCs

An important step in any safety engineering process is the derivation of safety requirements from analysis of system hazards. For example, the international safety standard IEC 1508 (ref. 2) defines the development of a system safety requirements specification designed to mitigate the identified system hazards. The system-level SVCs discussed in this paper correspond to system safety requirements.

The system-level SVCs are derived by treating the system as a "black box". This involves defining the system boundaries and the relevant system inputs and outputs. For example, the chemical factory hazard is ultimately tied to the vessel's actual temperature, which is reported by the external monitoring system. Both the monitoring system and the vessels are external to the chemical factory information system. The chemical factory information system receives reports from the vessel monitoring system as inputs and displays vessel temperature values as outputs. The system-level SVCs will be expressed in terms of these system inputs and outputs.

The derivation of the appropriate SVCs will be based upon results of the system hazard analysis. In the case of the chemical factory, the hazard analysis revealed that the system hazard may be the result of a corrupt or stale temperature value. SVCs are derived to mitigate these hazard causes.

In general, the system-level SVCs will include:

- functional correctness conditions;
- system and environmental assumptions required to carry out safety verification;
- constraints on variable system parameters or constants - typically expressed as mathematical inequalities, e.g., "X must be refreshed at a greater rate than Y".

The SVCs are typically constructed in an *ad hoc* fashion. Some of the limitations of an *ad hoc* approach to deriving SVCs are presented in the following sections.

<u>Functional Correctness:</u> The most obvious SVCs are "correctness" conditions necessary to mitigate the hazard. For the chemical factory, a SVC would be introduced to ensure that a displayed temperature value had been delivered in a timely and correct fashion. Another SVC would be introduced to ensure that stale temperature values are displayed as "unavailable". These type of SVCs are often captured in an *ad hoc* approach.

<u>Assumptions:</u> The introduction of SVCs will typically include assumptions about the system and the environment. For example, environmental conditions might be assumed such as the timely and correct delivery of the temperature value from the external vessel sensors to the system. Assumptions might be made about the ways in which the temperature display may be updated. These assumptions are often implicit in an *ad hoc* introduction of SVCs.

It is important that the underlying implicit assumptions are made explicit, especially if they are necessary for the mitigation of the hazard. The hazard-related assumptions should be included as additional SVCs.

<u>Temporal Constraints and Relations:</u> For a real-time system, some hazards will depend on the temporal ordering of events. For the chemical factory hazard, the display of invalid temperature values will depend on events such as the sensor's acquisition of data values, and the reception and display of temperature values by the information system. The processing of incoming temperature values will be concurrent with the monitoring of current temperature values for staleness.

Time-dependent hazards will lead to SVCs which place real-time constraints on the system. These SVCs will involve system constants that place limits on the system operations such as the minimum system propagation time for a temperature data value. Moreover, there will be dependencies between the temporal system constants. The system determination of a stale temperature value will be related to the maximum system propagation time for the temperature value. The relationships between the temporal constraints can be fairly subtle and not easily determined in an *ad hoc* approach. We suspect that the ability to systematically derive key temporal constraints and relations from hazard definitions may distinguish our approach most clearly from other existing methods.

<u>Mathematical-Style Reasoning</u>

Our approach uses a style of mathematical reasoning called "proof-by-contradiction". To prove an assertion "X", this style of reasoning begins with the introduction of a conjecture that X is not true, i.e., "not X". The argument proceeds by showing that "not X" inevitably leads to a contradiction. If this can be demonstrated, then we may conclude that "not X" is false – that is, X is true.

The task of showing that "not X" inevitably leads to a contradiction typically includes steps where the argument is "split" into multiple branches. Each branch of a split in the argument represents one of the cases in a case analysis. When the structure of the argument is viewed graphically, the splitting of some steps of the argument by case analysis has the effect of giving the graphical representation a "tree-like" appearance.

In the course of developing a proof by contradiction, we may introduce assumptions. The validity of the assertion proved in this manner with the aid of these assumptions will depend on the validity of the assumptions.

The next section of this paper describes how this mathematical style of reasoning may be adapted to serve as a means of generating SVCs.

Generating System-Level SVCs

A sketch of the analysis is provided for the chemical factory information system as a means of illustrating the process by which the SVCs are introduced.

Hazard analysis of the chemical factory information system reveals the existence of three time constants which have a direct relevance to the hazard:

- S1 - the maximum time required for a temperature value to be propagated through the chemical factory information system;
- S2 - the maximum time required for a temperature value to be propagated from the temperature sensors to the chemical factory information system via the external monitoring system;
- S3 - the maximum amount of time that the system will display the value in the absence of an external update before the system will set the displayed temperature to "unavailable".

S1 and S3 are software-level constants which denote upper bounds on the performance of the software system. S2 is an upper bound on the performance of an external system.

The discovery of SVCs begins with the conjecture that the identified system hazard has occurred at some particular instant of time. There are two main parts to the subsequent analysis. The first part of the analysis establishes a backward chain of causal relationships. These steps will include the introduction of three SVCs which will eliminate the display of a corrupt temperature value as a possible cause of the hazard. The second part of the analysis involves the introduction of additional constraints on the functionality of the system (i.e., more SVCs) that eliminates the possibility of a stale temperature value as the hazard cause.

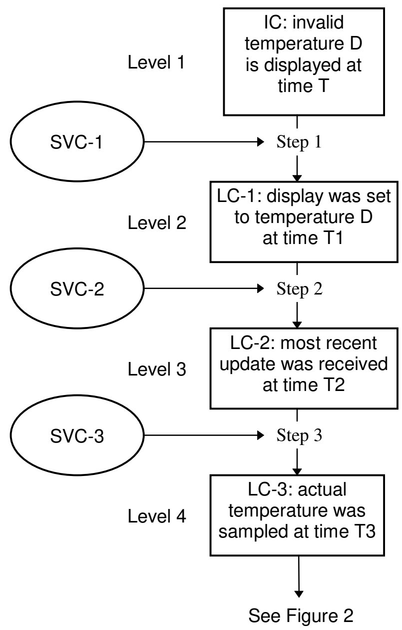

Figure 1: Graphical representation of the first three steps of the analysis.

Figure 1 provides a graphical representation of the structure of the first part of our analysis. The box at the top of Figure 1 represents the initial conjecture. The remaining boxes represent logical consequences (LC) of this initial conjecture. The ovals represent SVCs which are introduced as assumptions at various steps in the analysis. These assumptions are used along with the IC or LC of the "current" level to generate a LC for the next level of the analysis. Each level of the analysis is linked to the next lower level by an arrow. The arrow may be read as "implies". For example, the initial conjecture (IC), in combination with SVC-1, implies LC-1.

Initial Conjecture: The analysis begins with a conjecture that an instance of the hazard has occurred:

An invalid temperature, D, is displayed for vessel V at time T.

Based on the results of the hazard analysis presented earlier, the "initial conjecture" (IC) can be re-written in a more precise form as:

IC: invalid temperature D is displayed at time T.
The temperature, D, displayed for vessel V at time T has not been within MAX_DISP_TEMP _DIFF degrees of the actual temperature of the vessel at any time within MAX_DISP _TEMP_STALE milliseconds before time T.

Corrupt Temperature Value: As shown in Figure 1, the first three steps of the analysis each involve tracing backward from an event to its cause. In this respect, our approach is very much like the FTA process of tracing "backwards" from a hazard to its causes. Step 1 traces the occurrence of the hazard at time T (represented by the IC in the first level of analysis) backwards to an event at time T1 (represented by LC-1 in the second level of the analysis). Step 2 traces this event at time T1 to an earlier event at time T2. In turn, Step 3 traces this event at time T2 to an earlier event at time T3.

For this example, we have adopted the convention that upper case letters are used to denote specific instances (e.g., V, D, T, T1, T2, T3, T4). Lower case variables (e.g., v, d, t, t', t'') are used within the statement of a SVC for variables that are "universally quantified" by a "for all" operator.

In general, only some of the steps in the analysis will involve a relationship between an event and its cause. Some steps may be purely a matter of logical reasoning. The analysis will typically make use of logical laws (i.e., tautologies) as well as arithmetic laws. Although these laws may be cited in the written record of the analysis, they are not shown in the graphical representation.

Step 1: We suppose that the display part of the chemical factory information system is implemented by Commercial-Off-The-Shelf (COTS) hardware and software. For this illustrative example, we narrow the scope of our analysis to the application-specific, custom software which drives the COTS-based display subsystem. To this end, we introduce a high level assumption about the display subsystem which allows us to trace the cause of the hazard directly back to the application-specific, custom software.

SVC-1.
For all temperatures, d, times, t, and vessels, v, if d is displayed at time t as the temperature of vessel v, then there is some time t', t' <= t, when the temperature of vessel v was set to d and this was the most recent change made to the displayed temperature for vessel v.

Given this SVC, we can derive the LC-1 as a logical consequence of the initial conjecture.

LC-1: display was set to temperature D at time T1.
At time T1, T1 <= T, the temperature of vessel V was set to D and this was the most recent change made to the displayed temperature for vessel V.

Step 2: The displayed temperature value is the result of a system output which can be traced back to a system input. A second SVC is introduced to ensure that the displayed vessel temperature is the result of a temperature update from the external sensor monitoring system. Furthermore, the SVC ensures that the temperature update has been delivered correctly and within the time constraint, S1:

SVC-2.
For all vessels, v, displayed temperatures, d, and times, t, if the displayed temperature of vessel v is set to d at time t then at some time no earlier than S1 milliseconds before t the system received a report from the external sensor monitoring system that the temperature of vessel v is d.

Given SVC-2, we can derive the following as a logical consequence of LC-1:

LC-2a.
At some time T2, T2 < T1 and T1-T2 <= S1, the system received a report from the external sensor monitoring system that the temperature of vessel v is D.

Without loss of generality, this logical consequence may be refined into:

LC-2: most recent update received at time T2.
T2 is the most recent time before T1 when the system received a report from the external sensor monitoring system that the temperature of vessel V is D.

Step 3: The temperature update received by the system can be traced back to the vessel sensors. A third SVC is then introduced to ensure that the temperature update is correct, within a given tolerance, and has been delivered to the system within the time constraint, S2:

SVC-3.
For all vessels, v, displayed temperatures, d, and times, t, if the system receives a report at time t from the external sensor monitoring system that the temperature of vessel v is d, then at some time no earlier than S2 milliseconds before t the actual temperature of vessel v was within MAX_DISP_TEMP _DIFF degrees of d.

This SVC is used to derive the following logical consequence from LC-2:

LC-3: actual temperature was sampled at time T3.
At some time, T3, T3 < T2 and T2-T3 <= S2, the actual temperature of vessel V was within MAX_DISP_TEMP_DIFF degrees of D.

So far in our analysis, we have constructed a backward chain of events using symbolic names, T, T1, T2 and T3, to represent the times of these events. The order of these times is represented by the timeline in Figure 2.

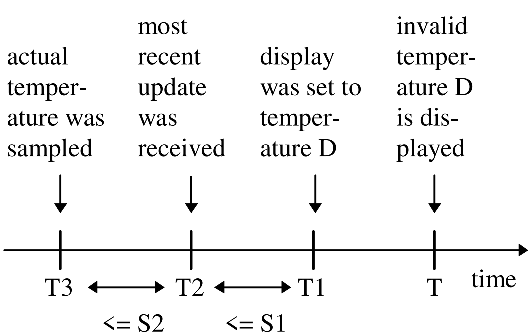

Figure 2: Timeline of Events

By the introduction of SVCs 1-3, the display of a corrupt temperature value has been eliminated as a possible hazard cause. The rest of the analysis involves the introduction of additional SVCs to remove the possibility of a stale temperature value as the cause of the hazard.

Stale Temperature Value: At this point, the analysis could be brought swiftly to a conclusion by the introduction of a SVC which asserts that the system will always change the displayed temperature to "unavailable" before the displayed value becomes stale. However, we contend that the phrase "before the displayed value becomes stale" is unacceptably vague. In particular, common sense suggests that the SVC(s) required to conclude this analysis should involve one or more explicit references to the value of S3.

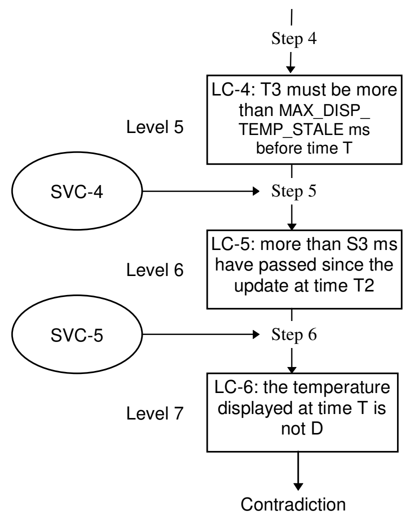

Figure 3: Graphical representation of the second part of the analysis.

As defined earlier in this paper, S3 is a constant of the software system for the maximum amount of time that a value may be displayed before it is considered stale. We can imagine that S3 is used in the software implementation of the system to set a timer when an update is received. If the timer is not reset by a subsequent update, then eventually the software will initiate an action to cause the displayed temperature to be changed to "unavailable".

Hence, the value of S3 is a key consideration in the mitigation of this hazard. It is easy to construct scenarios where "too small" or "too large" a value for S3 may cause an occurrence of the hazard. Common sense leads us to expect that S3 is related to the "requirements-level" system constant MAX_DISP_TEMP_STALE, but the precise relationship between the two values may not be so obvious.

Thus, the remainder of our analysis focuses on the problem of generating additional SVCs which constrain the value of S3, as well as the behaviour of the system with respect to S3. See Figure 3 for a graphical representation of the second part of the analysis.

Step 4: In the first part of the analysis, we have narrowed the cause of the hazard occurrence to the situation where a displayed temperature has become stale. This can be seen more clearly if LC-1 and LC-2 are used to derive the inequality T2 < T, and this inequality is, in turn, used to derive the following logical consequence from LC-3:

> **LC-4a.**
> At some time, T3, T3 < T, the actual temperature of vessel V was within MAX_DISP_TEMP_DIFF degrees of D.

We know from the initial conjecture, IC, that the actual temperature of vessel V was not within MAX_DISP_TEMP_DIFF degrees of D at any time within MAX_DISP_TEMP_STALE milli-seconds prior to T. So, in light of LC-4a, T3 must be more than MAX_DISP_TEMP_STALE milliseconds prior to T. This reasoning yields LC-4:

> **LC-4: T3 must be more than MAX_ DISP_TEMP_STALE ms before T.**
> At some time, T3, T - T3 > MAX_DISP_TEMP _STALE, the actual temperature of vessel V was within MAX_DISP_TEMP_DIFF degrees of D.

Step 5: A new SVC is introduced to constrain the value of S3. MAX_DISP_TEMP_STALE is the maximum amount of time that a value may be displayed before the value is considered to be stale. Similarly, S3 is the maximum amount of time the system may display a temperature value before the value is considered to be stale, but as measured from the time the value is first received by the system. The value of S3 must be less than MAX_DISP_TEMP_STALE to take into account the time required for the temperature value to reach the system from the external monitoring system. Since S2 is the maximum time allowed for a temperature value to reach the chemical factory information system, the following SVC is proposed:

> **SVC-4**.
> MAX_DISP _TEMP_STALE > S2 + S3

This SVC is used to derive (by transitivity of ">") the following logical consequence from LC-4:

> **LC-5a.**
> T - T3 > S2 + S3

From LC-3, we know that T2 - T3 <= S2. This relation is used, along with various rules of arithmetic, to derive the following logical consequence from LC-5a:

> **LC-5: more than S3 ms has passed since the update at time T2.**
> T - T2 > S3

Step 6: We can informally assume at this point that the receipt of an update at time T2 initiates a process that will cause the displayed temperature to be changed to "unavailable" if a subsequent update is not received before the displayed temperature becomes stale. If a subsequent update is received in time, the display is updated to the new value. This leads to the fifth SVC:

> **SVC-5.**
> For all vessels, v, and times, t and t', if time t is the most recent time that an update for vessel v at temperature d was received prior to time t', and t'-t > S3, then at some time, t'', such that t+S1 < t'' <t', the temperature value displayed for vessel v shall have been set to "unavailable" or shall exhibit some value other than d.

Considerable care went into the formulation of SVC-5 to ensure that it is both sufficiently general (i.e., constraints on the actual implementation are minimal) and practically feasible (i.e., there is likely to be a practical implementation of this SVC). This new SVC is used to derive the following logical consequence of LC-5:

LC-6a.
At some time T4, T2+S1 < T4 < T, the temperature value displayed for vessel v shall have been set to “unavailable” or shall exhibit some value other than D.

We know from LC-2a that T1-T2 <= S1 and, from LC-6a that T2+S1 < T4. Using rules of arithmetic reasoning for inequalities, we can use the inequality T1-T2 <= S1 as justification for replacing S1 by T1-T2 in T2+S1 < T4 to obtain T2+(T1-T2) < T4. In its simplified form, T1 < T4, this result clearly shows that T4 must be some time after T1. Therefore, we can derive the following logical consequence of LC-6a:

LC-6: the temperature displayed at time T is not D.
At some time T4, T1 < T4 < T, the temperature value displayed for vessel v shall have been set to “unavailable” or to some value other than D.

LC-6 contradicts LC-1, which states that the most recent change to the displayed temperature value occurred when it was set to D at time T1.

QED

Discussion: The result of the above analysis is the discovery of five distinct safety verification conditions. Although some of these SVCs might have been anticipated, we believe that it would be difficult to determine the precise details of these conditions without carrying out an analysis such as the one described above.

For example, SVC-5 states that the system shall update the temperature display for vessel V as “unavailable” within S3 milliseconds of the most recent update for vessel V. SVC-5 also states that the update should occur only after S1 milliseconds had elapsed since the most recent update. The choice of S1 as the lower bound is necessary if SVC-5 is to be used to derive the contradiction, LC-6, from LC-5. If a value smaller than S1 is chosen, LC-6 would then allow the temperature of vessel V to be set to “unavailable”, or to some other value other than D, at some time before time T1. As a result, there would be no contradiction. We suggest that the necessity of a lower bound is not particularly obvious – and its specification may easily have been overlooked if an *ad hoc* process had been used to generate the SVCs for this example.

Our approach is not a purely “mechanical” method for the generation of SVCs. Some inspiration is required to formulate each of the SVCs. But it is generally better to depend upon a systematic process which requires many “small” inspirations than an *ad hoc* approach which depends on a few “big” inspirations.

We have found that the use of inequalities involving variables that represent instants of time (e.g., t, t’ and t’’) is satisfactory as a means of specifying temporal relationships. An alternative, which we have not yet investigated in this context, is the use of temporal logic notation.

Though we have found that an informal, but rigorous, style of reasoning to be sufficient for the derivation of the SVCs, it may be worthwhile validating the analysis by means of a formal verification technique. For example, an earlier version of the analysis presented in this paper contained errors that invalidated the final rigorous argument. These errors included a version of SVC-5 that contained a “loophole” that would have allowed a stale temperature value to be displayed if the system received but failed to propagate a temperature value. These errors were sufficiently subtle to escape detection by three external reviewers of the earlier version of this paper.

Most of the reasoning required by this example was a matter of arithmetic reasoning about inequalities. The reader may try his or her hand at the derivation of LC-5 from LC-5a and the inequality T2-T3 <= S2 to get a sense that this kind of reasoning, though error-prone, is nothing more than “high school level mathematics”. A bit of predicate logic reasoning was implicitly used in several steps, such as the refinement of LC-2a into LC-2. To ensure greater confidence in the analysis and the resulting SVCs, it may be useful to perform the analysis with the aid of a formal verification tool.

Generality

The two key elements of our approach are: (1) “proof-by-contradiction” style reasoning and (2) the introduction of SVCs during the reasoning process to “steer” the proof-by-contradiction towards the closure of each branch of the proof. Our style of representing the derivation of SVCs graphically, as demonstrated in Figures 1 and 3, may also be applied generally to other applications of our method.

Related Work

The refinement of a hazard definition into a set of SVCs is similar in some respects to FTA. FTA is often used during hazard analysis to uncover and organize hazard causes. FTA begins with the identified hazard and then works backward to uncover all possible causes. The intermediate events that cause the hazard are combined using logical operations such as “AND” and “OR”. The nodes of the fault tree can be used as the basis for the SVCs. For example, software safety requirements were derived from an FTA of a Magnetic Stereotaxis System (ref. 8).

However, FTA is limited to a form of propositional reasoning where each branch in the tree corresponds to a disjunction or a conjunction. Our approach involves “proof-by-contradiction” style of reasoning that makes informal use of quantification and rules of reasoning based on predicate logic.

SFTA also makes use of “proof-by-contradiction” style reasoning. SFTA is essentially FTA performed at the source code level. SFTA begins by assuming a hazardous output from a given line of source code. The hazard causes are then traced backwards through the code with the help of language templates. The templates are based on the semantics of the programming language and determine the various ways a code statement can contribute to the hazard or to an intermediate event. The analysis continues until a contradiction is reached.

Unlike SFTA, our approach is not tied specifically to templates based on the syntax and semantics of a programming language. Whereas SFTA is intended to be used as means of verifying the source code with respect to a defined hazard, our approach is meant to support the derivation of system SVCs.

Summary

The approach illustrated by the example in this paper provides an alternative to an *ad hoc* approach to the discovery of SVCs. The result of our earlier efforts to write SVCs described earlier in Section 2 lacked the precision of the five conditions that we later discovered as a result of performing the analysis sketched in Section 3. In addition to the principle objective, namely, the discovery of SVCs, our approach yields a rigorous argument that may be used to increase confidence in the safety of the system. Of course, the validity of this argument depends on showing that the implementation of the system satisfies the SVCs.

Acknowledgments

The work described in this paper is a result of a collaborative industry/university research project sponsored by the BC Advanced Systems Institute, Raytheon Systems Canada Ltd., and MacDonald Detwiller. The authors are grateful to Cerina Koster and Jerry Grummer of Raytheon Systems Canada Ltd. for their comments on a draft version of this paper.

References

1. Department of Defense, “Military Standard 882C: System Safety Program Requirements”, 1993.

2. International Electrotechnical Commission. “Draft International Standard IEC 1508: Functional Safety: Safety Related Systems”. Geneva, 1995.

3. W. E. Vesley, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. “Fault Tree Handbook”. NUREG-0942, U.S. Nuclear Regulatory Commission, 1981.

4. Nancy G. Leveson, Steven S. Cha, and Timothy J. Shimall. “Safety Verification of Ada Programs using software fault trees”. IEEE Software, vol. 8, no. 7, pp. 48-59, July 1991.

5. Bruce Elliott and Jim Ronback. “A System Engineering Process For Software-Intensive Real-Time Information Systems”. Proceedings of the 14th International System Safety Conference, Albuquerque, New Mexico, August 1996.

6. Nancy G. Leveson. “Safeware: System Safety and Computers”. Addison-Wesley, 1995.

7. http://www.cs.ubc.ca/formalWARE/

8. John C. Knight and Darrell M. Kienzle. “Preliminary Experience using Z to Specify a Safety-Critical System”. Department of Computer Science, University of Virginia, Technical Report.

9. Ken Wong. M.Sc. Thesis. Department of Computer Science, University of British Columbia, 1998.

Biographies

Jeffrey Joyce, Raytheon Systems Canada Ltd., 13951 Bridgeport Road, Richmond, BC, V6V 1J6, telephone - (604) 279-5721, fax - (604) 279-5982, email - jjoyce@west.raytheon..com

Dr. Joyce is a Research Scientist at Raytheon Systems Canada Ltd. as well as an Adjunct Professor at The University of British Columbia.

Ken Wong, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, V6T 1Z4, telephone - (604) 822-4912, fax - (604) 822-5485, email - kwong@cs.ubc.ca

Ken Wong is currently completing his master's thesis at the University of British Columbia in the Department of Computer Science. His research interests include software safety and software architectures.