延伸正規方法 SpecTRM-RL 以進行安全分析
Extending Formal Method SpecTRM-RL for Safety Analysis

史龍安　　范金鳳
Lung-An Shih　　Chin-Feng Fan

元智大學資訊工程學系
Yuan-Ze University, Chung-Li, Taiwan
csfanc@saturn.yzu.edu.tw

摘要

本論文延伸 Nancy Leveson 的最新正規方法 SpecTRM-RL，我們由 SpecTRM-RL 規格中，衍生了狀態轉換圖及 UML 時序圖以增進此方法的可讀性。我們又在狀態圖及 SpecTRM-RL 上添增故障情況（failure conditions），並根據此延伸的規格並發展了一套有系統建構故障樹的方法，以進行安全分析。我們的目的在提昇電腦控制系統的軟體安全性。我們以火車過平交道案例來說明此方法的可行性及有效性。

關鍵詞：SpecTRM-RL，故障樹，安全分析

Abstract

This research extended Nancy Leveson's newest formal specification method SpecTRM-RL. From SpecTRM-RL, we generated induced state transition diagrams and UML sequence diagrams to improve the understandability of the method. Moreover, we added failure conditions in the state transition diagrams and SpecTRM's AND/OR tables. Based on the proposed extension, we developed a systematic fault tree synthesis mechanism to perform safety analysis. Our goal was to enhance software safety in computer-controlled systems. A railroad crossing case was used to demonstrate the feasibility and effectiveness of the proposed method.

Keywords: SpecTRM-RL, fault trees, safety analysis.

一、序論

電腦控制系統或嵌入式系統，常涉及安全議題，對這些系統而言，如何制定軟體需求規格，並能進一步驗證其規格是安全的，是一項重要的課題。安全關鍵系統的正規規格（即數學為基礎的方法）常用來描述其軟體需求及設計。現行軟體正規方法眾多，含圖形方式（如 Statechart [4]，Petri Net [9]）、模型為主（Model-based）的方式（如 Z [1]，VDM[6]）、以及性質為主（Property-based）的方式（如 Temporal Logic [10]等）。SpecTRM-RL [12]是 Nancy G. Leveson 所發展的一套最新的正規化規格語言。Leveson 被譽為軟體安全分析的鼻祖，她所提出的安全相關方法具有相當的影響力。Leveson 的 SpecTRM-RL 是建構在她以往的 RSML [5]規格語言，及 Intent [8]的需求規格架構上。其特色是以控制理論（control theory）為基礎，進行初步安全分析（preliminary safety analysis），在系統規格中制定限制項目（constraints），並在執行時遵守這些限制，經由回饋資訊 (feed back loop)，做出適當的調整以確保系統安全。

因 SpecTRM 仍在發展中，尚有不足的地方，例如：我們認為安全分析方面不應只在事前分析，開規格後應該也需要進行安全分析。而在系統運作過程中，只有範圍、時間、數值等限制（constraints）來維持系統安全是不夠

的，我們建議應該增加斷言（assertion）[11] [13]即變數間的關係式。

本研究的目的針對電腦控制系統或嵌入式系統，延伸 SpecTRM-RL 方法，使其可進行進一步安全驗證。我們的方法包括以下步驟：

1. 根據 SpecTRM-RL 模型建構狀態轉換圖
2. 根據 SpecTRM-RL 模型畫出 UML 時序圖
3. 畫出增添式狀態轉換圖（augmented state transition diagram）
4. 延伸 SpecTRM-RL 成為增添式 SpecTRM 模型（augmented SpecTRM model）
5. 系統化的建構故障樹
6. 以 UML 時序圖表達故障情境

我們的方法延伸 SpecTRM-RL，應用 UML 時序圖可提升 SpecTRM-RL 的可讀性並明確表達出子系統/元件間之互動；增添式狀態轉換圖增加了錯誤的狀況，有利於驗證，能有系統的分析出可能的不安全情境。

圖 1　SpecTRM [12]

二、相關背景研究

2.1 SpecTRM

SpecTRM[12] (Specification Tools and Requirements Methodology)是 Leveson 根據她的 Intent Specification[7]的規格架構來發展的一套軟體規格工具及方法論。而 Intent Specification 則是一套完整的軟體規格架構，它分為 7 層（levels）包含高層級的需求和安全限制（包含危險分析 hazard analysis）至元件規格的模型、編碼和運算工作。而 SpecTRM-RL 是建構在 Level 3，用來表達黑箱行為。圖 1 為 SpecTRM 方法中的系統圖、限制圖和 AND/OR 表。

2.2 故障樹（Fault Tree）

故障樹分析（Fault Tree Analysis）是一種廣泛使用在飛航、核能、電子工業的系統安全分析方式，目的在於分析造成危險及危害的原因。它由一不安全的事件往前推導其原因，而原因間使用 AND、OR 閘來表示造成危險原因的組合至元件層次，其示意圖如圖 2 所示。長方形為可展開的事件，菱形為不再展開的事件，圓形為終結的原因或元件層次的故障。

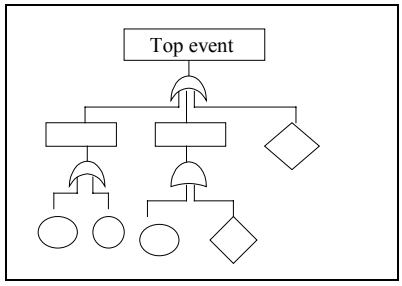

圖 2 故障樹示意圖

三、SpecTRM 的延伸

我們延伸的方法首先依 SpecTRM-RL 畫出每個物件的狀態轉換圖（state transition diagram）和 UML 時序圖，再者在狀態圖及 AND/OR 表中加上 ***失效條件（failure conditions）*** 成增添式狀態轉換圖和增添式 SpecTRM-RL 模型，最後以有系統的方式合成故障樹，以進行安全分析。本研究的步驟如圖 3 所示。各步驟詳細於下面各節。

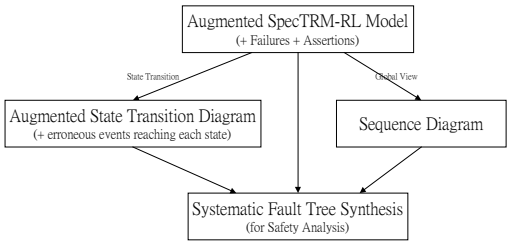

圖 3 我們的方法

3.1 根據 SpecTRM-RL 來建立狀態轉換圖及 UML 時序圖

我們首先依 SpecTRM 的規格來畫其輸入裝置、輸出裝置、及控制器的狀態轉換圖（state transition diagram），包含其間訊息的傳遞與接收，如圖 4 所示。有的狀態間的順序無法完全推測出，得加上主觀專業知識（domain knowledge）來完成。

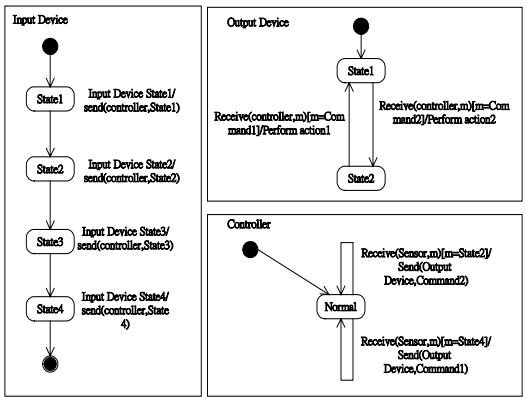

圖 4 推導出的 SpecTRM-RL 通用狀態轉換圖

接著我們依 SpecTRM-RL 的模型畫出 UML 的時序圖以便於視覺上即能了解子系統間的互動狀況。我們將時序圖分為輸入裝置與控制器間的互動（及送狀況/事件訊息），和控制器與輸出裝置間的互動（即輸出指令）。其方法如圖 5，6 所示。

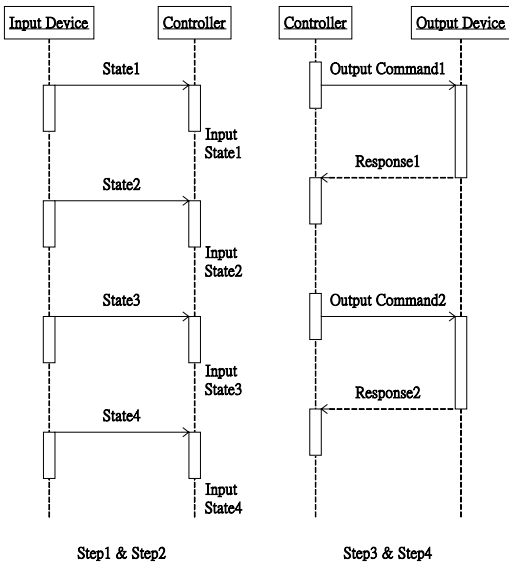

圖 5 合併前的模版圖

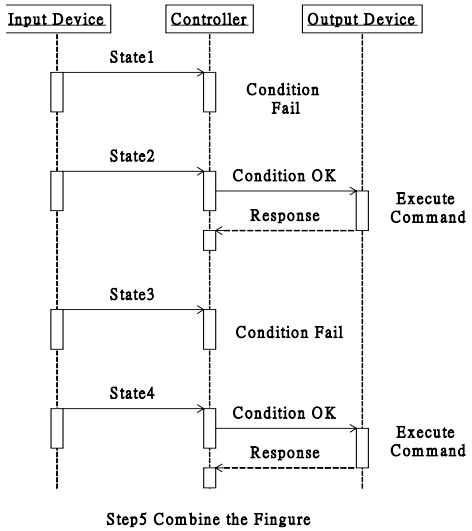

圖 6 合併後的完成圖

3.2 建立增添式狀態轉換圖及增添式 SpecTRM-RL 模型

為了利於後續故障樹的建立及分析，我們

在狀態轉換圖上添加的失誤的原因（failure conditions）。我們有 2 個前提：

1. 我們是針對被電腦控制的子系統（Controlled subsystems）畫出其故障情況。
2. 假設子系統每一狀態皆由單一個 (unique) 觸發訊息或事件決定。

我們的方法如下（如圖 7 所示）：

1. (a) 實線表示是正常情況的狀態轉換。虛線表示錯誤的狀況。圖 7 中（a）表示正常狀況。
2. 增加（b）該發生卻沒發生的狀況。子系統原先在 Q，當收到不是 P（¬P）的訊息—應該轉換到另一個狀態時，子系統卻停留在 Q。
3. 增加（c）不該發生卻發生的狀況。因 ¬P∧Q 的虛線表示沒收到 P 的訊號子系統卻自動進入 Q 的錯誤狀況（c）。

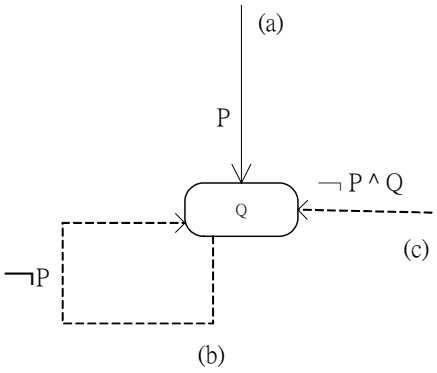

圖 7 增添式狀態轉換圖

同樣地，我們可在 AND/OR 表中增加故障情況描述。Critical failure 指該發生卻沒發生的錯誤情況，而 Spurious signal 指不該發生卻發生的錯誤狀況，No Action 表示 Output Device 在 Q 狀態並且當收到 Input Device 其他狀態時，應該送出根據其他狀態的指令，但卻沒有動作。根據圖 7 我們畫出如圖 8 示意圖。

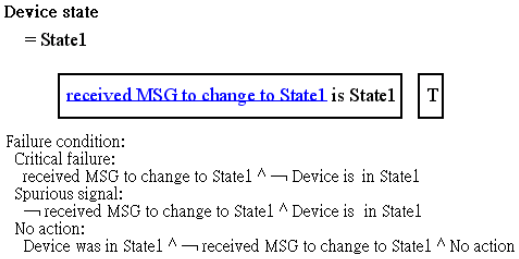

圖 8. Augmented AND/OR 表 (加上 Failure)

3.3 系統化的建構故障樹

我們的方法與 Leveson 方法的主要差異在於我們主要發展了一套由 SpecTRM-RL 產生故障樹的方法。雖然 Leveson 極力批評 event-chain 式安全分析方法[8]，然而 event- chain 方法如故障樹等，仍是延用最多，最多人熟稔的系統安全分析方法，我們相信可與 SpecTRM 的方法合併，並不衝突。

電腦控制系統包含主動及被動式系統，主動式的系統，不會受外在的訊息控制，例如：火車。被控制的系統，接受外在的訊息控制，例如：柵欄。我們的故障樹建構方法如下：

1. Top Event 先確認危險情況的狀態所在位置，先判斷是主動物件還是被動物件，主動物件則從其時間點往回推上一個狀態；而被動物件則根據增添式狀態轉換圖，如圖 7 中可看出有 3 種錯誤的原因，該作沒做（Critical failure），不該做而做（Spurious signal），和錯誤地維持現狀（No action），再進行步驟 2。
2. 針對下列錯誤推測前因：(a) 訊息錯誤發生的原因，可以從時序圖判斷，傳送端和接收端都有可能出錯，要視錯誤情況而定。(b) 硬體和狀態轉換的錯誤，可以參考我們增添式狀態轉換圖。(c) 軟體錯誤則可以經由增添式 AND/OR 表中的錯誤情況（Failure）來找原因。
3. 重複 1, 2 的步驟來建構故障樹。

我們所建構出的故障樹的模版（template）如圖 9 所示。線段上的文字為說明或註明查何種圖表。

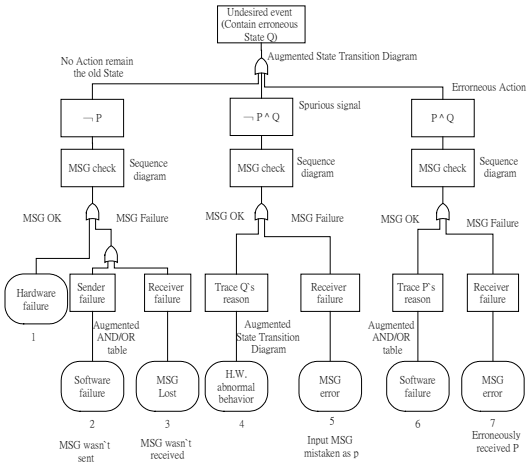

圖 9 系統化故障樹之合成（Synthesis）

其故障情境可由 UML 時序圖表達。如圖 10 所示。

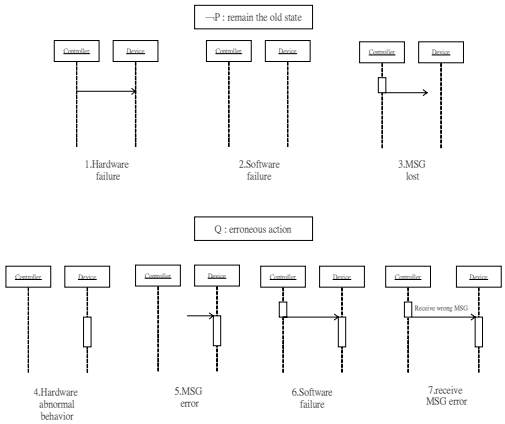

圖 10 時序圖表達故障情境

四、應用案例

4.1 火車過平交道應用

我們以單一火車通過平交道的應用案例來說明我們方法的可行性及有效性。其基本 SpecTRM-RL 模型如圖 11 所示，其推導出的狀態轉換圖如圖 12 所示。本系統包含火車、柵欄、和控制軟體三部分。火車有 Traveling、Approach、Ingate 和 Exit 四個順序狀態接送訊息通知 Controller，Controller 則送訊號控制 Gate 的上升及下降。

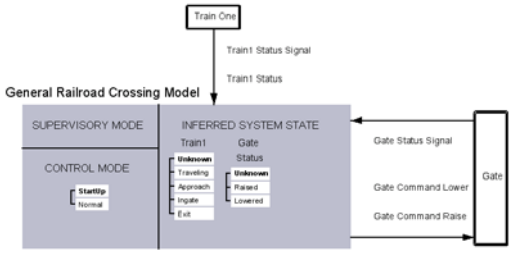

圖 11 火車過平交道 SpecTRM-RL 系統圖

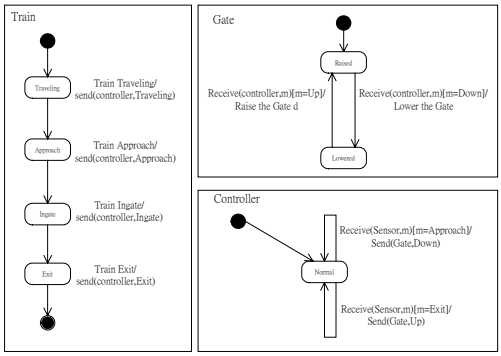

圖 12 火車過平交道狀態轉換圖

圖 13 為相關的 AND/OR 表。我們根據此表畫出互動時序圖如圖 14。

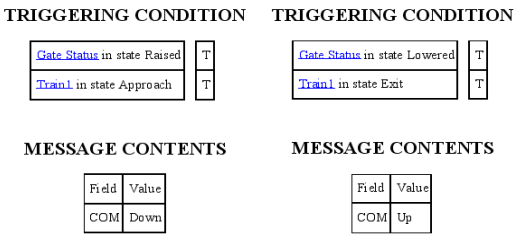

圖 13. 控制柵欄指令之 AND/OR 表

接著我們根據 3.2 節的方式畫出增添式態轉換圖(圖 15,16)及增添式 AND/OR 表(圖 17)，兩者皆包含前述錯誤的狀況（failures）。

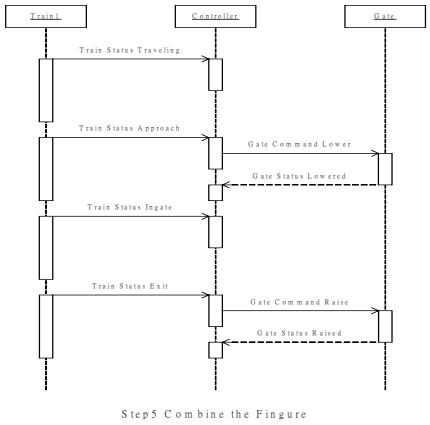

圖 14 火車過平交道的合併時序圖

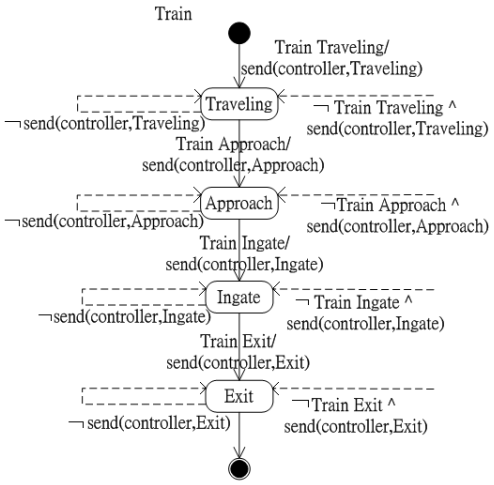

圖 15 增添式火車狀態轉換圖

接著我們以意外事件（top event）（Gate=Raised）∧（Train=Ingate）為例，根據圖 9 方式產生故障樹如圖 18 示。

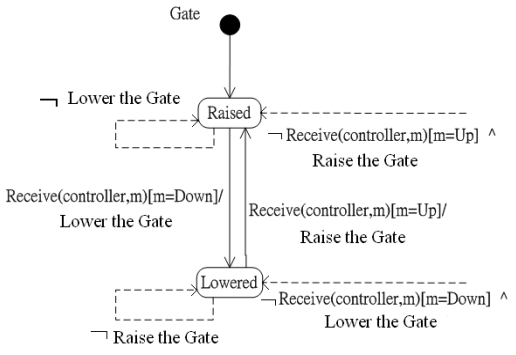

圖 16 增添式柵欄狀態轉換圖

在圖 18 中我們可以發現主要有 3 種類的錯誤：

1.硬體錯誤（Hardware failure）：

編號包含 1,2, 9, 10, 11, 15 事件。（例如其中編號 1 表示柵欄硬體故障）

2.軟體控制錯誤（Software controller failure）：

編號包含 5,6,13。（例如其中編號 5 表示控制器判斷錯誤）

3.訊息傳遞錯誤（MSG failure）：

編號包含 3,4,7,8, 12,14, 16。（例如其中編號 3 表示柵欄沒收到訊息的錯誤）

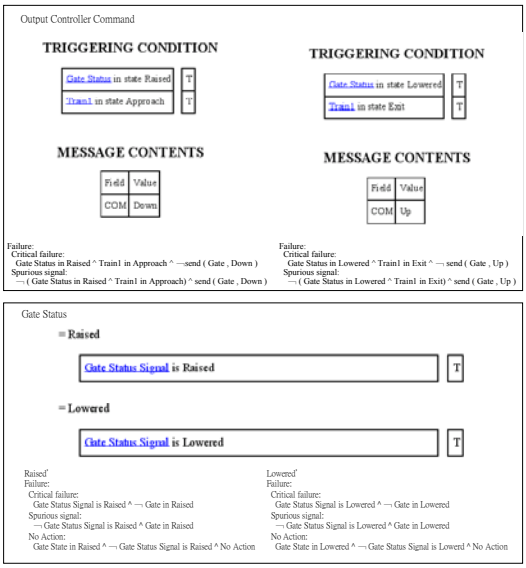

圖 17 增添式 SpecTRM-RL 模型

假設我們有興趣的是軟體的故障原因編號 5，軟體判斷錯誤未送出訊息，則其情境的時序圖如圖 19 示。

經由此故障分析可分析出可能的不安全處，可重新設計以增進安全。例如當我們發現可能發生軟體錯誤時，可加上多元化設計（N-diversity version）或硬體 interlock。而硬體的錯誤時，可新增硬體結構的一些限制或多元化設計（design diversity），以確保系統的安全性。

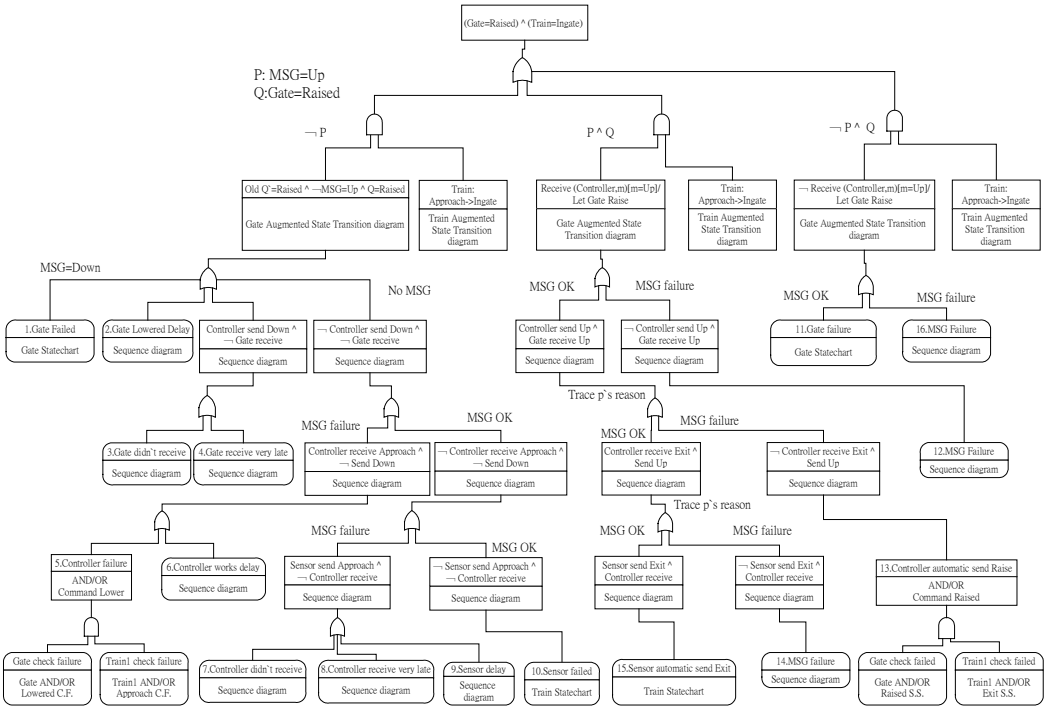

圖 18 火車過平交道之故障樹

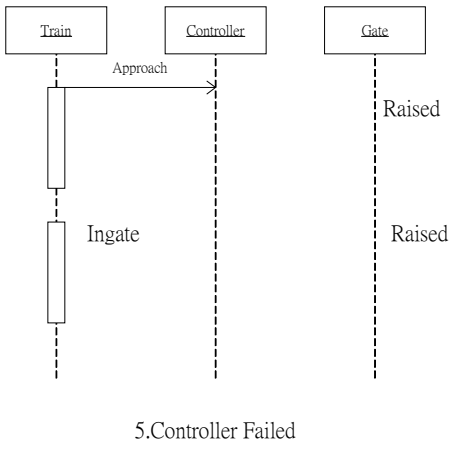

圖 19 軟體錯誤情況

4.2 限制(Constraints)外加斷言(Assertion)

除了上述靜態事前的故障樹安全分析法外，SpecTRM-RL 的限制（constraints）提供執行時的檢查。SpecTRM-RL 所顯示的限制，多數為第一變數，裝置在時間範圍等上的限制。我們以為尚不足以確保系統執行時的安全性。，我們建議增加斷言（Assertion）[13]。斷言可表達變數之間的預期正常關係，在執行時電腦可以檢查此類預期正常關係是否在。若不存在，應警示操作員。火車的例子較簡單，斷言的需求不很明顯。以另一常被引用的規格案例蒸汽鍋爐（Steam Boiler）[2]而言，此案例中有幫浦（pump），加熱器（heater），放水閘門（water valve），釋壓閘門（release valve），而系統中有些程序變數（Process Variable），例如溫度、水位、壓力。我們建議可能的斷言可分為裝置與裝置之間（Device-Device relation），裝置與程序變數之間（Device-Process variable relation），和程序變數和程序變數之間（Process variable-Process variable relation）等類型，以下為部分範例。

1. 裝置之間（Device - Device relations）：
   例如：水位太低開幫浦，並關放水閘門。
2. 裝置—程序變數之間（Device - Process variable relations）：
   例如：幫浦打開 → 水位上升
   釋壓閘門打開 → 壓力下降
3. 程序變數—程序變數之間（Process

variable – Process variable relation）：

例如：目前鍋爐中的水量＋幫浦加入的水量—放水閘門的出水量＝下一刻的鍋爐中的水量

以上的斷言例子，多少都涉及時間上的動作與結果的效應。斷言亦可有其他多種類型，可見 [13]的分類。電腦可在執行時檢查系統是否符合預先設定的斷言的關係及限制範圍數值，若不符合，則執行預設的處理方式（Exception handling）或警示操作員。執行此檢查斷言及限制的電腦可以與控制部分（controller）是相同的，亦可以是獨立的另一個斷言處理器（assertion processor）。如此執行時限制加上斷言的檢查可彌補事前安全分析的可能不足之處。

五、結論

本研究延伸了 Leveson 的 SpecTRM-RL，增加了狀態轉換圖和時序圖，並在轉換圖及 AND/OR 表加了錯誤到達此狀態的可能轉換條，我們並發展了有系統的建構故障樹的方法。我們的圖形方式(如 Sequence diagrams, augmented state transition diagrams) 增進 SpecTRM-RL 的可讀性，並可進行安全驗證，以期提昇系統的安全性。未來擬將發展出的方法及步驟加以自動化並應用於實際案例上以評估具體的功效。

六、參考文獻

[1] J. R. Abrial, “The specification language Z : syntax and semantics,” *Programming Research Group,* Oxford, April 1980.

[2] J. R. Abrial, et al., *Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control,* LNCS 1156, Springer-Verlag, Oct. 1996.

[3] M. Cepin and B. Mavko, “Fault tree developed by an object-based method improves requirements specification for safety-related systems, ” *Reliability Engineering and System Safety,* vol. 63, pp. 111-125, 1999.

[4] D. Harel, “ Statecharts: a visual formalization for complex systems,” *Sci. Comput. Program.* Vol. 8 , pp. 231-274 , 1987.

[5] P.E. Heimdahl and N. G. Levson, “Completeness and consistency in hierarchical State-Based Requirements,” *IEEE Transactions on Software Engineering,* Vol. 22, No. 6, June 1996.

[6] Cliff B. Jones, *Systematic Software Development Using VDM*, Prentice Hall, January 1990.

[7] N. G. Leveson, “Intent specification: an approach to building human-centered specifications,” *IEEE Transactions on Software Engineer,* Vol. 26, No. 1, January 2000.

[8] N. G. Leveson, “A new accident model for engineering safer systems,” Proc. of International System Safety Society Conference, August 2002.

[9] J.L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981.

[10] N. Rescher and A. Urquhart, *Temporal Logic*, Springer-Verlag, Library of Exact Philosophy, 1971.

[11] D. S. Rosenblum, “Towards a Method of Programming with Assertions,” ACM Proceedings of the 14th international conference on Software engineering, June 1992.

[12] Software Engineering Corporation, “SpecTRM User Manual,” 2001.

[13] Swu Yih, Jeff Tian, “Developing and checking prescriptive specifications for safety improvement,” *Microprocessors and Microsystems,* vol. 21, pp. 587-594, 1998.