特 輯

시스템 信賴性(System Reliability)

千柳植, 金東珠, 尹炳欽, 韓永哲
三星一지.티.이 通信株式會社

〈 要 約 〉

시스템의 좋고 나쁨의 구별에 決定的인 役割을 하는 信賴性에 對하여 설명하였다. 信賴性은 단순히 正確性보다는 缺陷包容性에 依存함이 크며 主 統制 및 제어裝置가 컴퓨터인 시스템에 있어서는 設計時에 Software에서의 缺陷包容設計의 重要性을 강조하였다.

缺陷包容設計의 一般的인 方法과 주요부분의 二重化에 對하여 컴퓨터의 二重化方式과 二重化의 長短點을 살펴보고, 缺陷包容設計의 한 예로써 GTK-500 EPABX를 설명하였다.

〈 Abstract 〉

The value of a system is highly dependent upon its reliability. Reliability means not merely correctness but means fault tolerance of the system. This paper emphasizes software fault tolerance in design stage especially in case of computer controlled system.

The general method of fault tolerance design especially including dual computer system and its advantage and disadvantage was introduced. Finally for example of fault telerance design we would like to present our GTK-500 EPABX.

1. 序 論

아무리 좋은 機能을 가진 시스템이라도 使用도중에 故障이 자주나면 좋은 시스템이라고 하기 어렵고 경우에 따라서는 한 번의 故障이 致命的인 결과를 초래할 수도 있다. 그러나 完全無缺한 시스템은 있을 수 없으며 그것에 가까와지면 相應하는 代價(cost)를 치러야 한다.

오늘날 우주科學, 發電, 電話, 生産過程등의 分野에 統制하게 되어, 이제 종합적인 면에서의 시스템 信質性이 중요한 의미를 갖게 되었다. 각각의 部品(component)이나 組立品(assembly)의 信賴性은 물론 그것이 시스템 전체에 어떠한 영향을 미치게 되는가를 고려하여 가장 經濟的인 시스템을 구성하는 것이 필요하다. 더욱이 主제어裝置인 컴퓨터에 對한 依存度가 높으므로 컴퓨터 시스템 자체의 信賴性이 중요한 問題가

된다.

컴퓨터는 hardware와 software로 구별되며, Hardware의 경우엔 아무리 高信賴度部品을 使用하여도 故障이 일어나는 確率을 zero로 할 수 없기 때문에 software는 그 자체의 正確性(correctness)보다는 hardware의 缺陷包容(fault telerance)이 더욱 중요하다. 시스템 設計時에 software設計者는 컴퓨터의 hardware는 물론 시스템 전체의 信賴性 目標를 위하여 助言을 하여야 하고 缺陷包容設計(fault tolerance design)를 하여야 한다.

여기서는 信賴性에 관한 一般的인 사항을 기술하고, 高信賴性 달성을 위하여 필요한 設計 즉 缺陷包容設計에 對한 설명을 한 후 예로써 GTK-500電子式 構成交換機에 있어서의 주요부분의 二重化와 software의 缺陷包容設計에 의한 信賴度 향상에 對하여 설명하였다.

2. 一般 및 公式

1) 品質(Quality)과 信賴性(Reliability)

品質과 信賴性은 흔히 혼동되는 경우가 있다. 前者는 시스템을 구성할 때 까지의 各 구성要素들의 缺陷의 적음을 뜻하고 後者는 시스템의 수명期間(life cycle time)에서의 故障의 적음을 말한다. 品質이 좋으면 信賴性이 좋아지는 등의 관계가 있어서 品質은 信賴性의 必要條件은 되지만 充分條件은 되지 못한다.

2) 正確性(Correctness)과 信賴性(Reliability)

시스템에서 正確性과 信賴性은 구별된다. 아무리 시스템 設計를 正確하게 하고 設置를 잘하여도 부분적으로는 hardware의 故障이 일어나므로 이것을 對備하여 缺陷包容設計를 하지 않으면 시스템의 信賴性을 기대할 수 없다. 대부분의 缺陷은 hardware에서는 部品범주에 해당하므로 처음 設計時에 software에서 缺陷包容設計를 잘하지 않으면 信賴性을 달성할 수 없다.

3) 波及效果(Domino Effect)

시스템에서 어떠한 過程(process)에서의 하나의 缺陷이 시스템 전체의 故障으로 波及되는 경우가 많은데 이것을 domino effect라고 한다. 그림 1에서, 過程 1에서 缺陷이 일어나면 가장 最近의 복구지점 4로 되돌아가서 다시 進行하게 되므로 다른 過程엔 영향을 미치지 않는다.

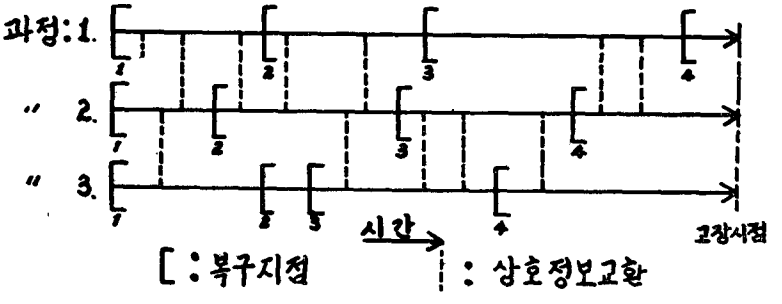

그림 1. 파급 효과

(說明의 편의상 처음의 복구지점에서 복구되는 것으로 가정하였다.) 過程 2에서 일어나면 복구지점 4로 되돌아가면 되나 過程 1에도 영향을 이미 미쳤기 때문에 복구지점 3(過程 1의)로 되돌아가야 한다. 過程 3에서 일어나면 出發지점으로 되돌아가야 하는데 이것은 시스템에서 볼 때 심각한 사태가 아닐 수 없다.

4) 시스템수명과 故障率

시스템의 수명期間동안의 故障率은 그림 2와 같은 그라프를 그린다.

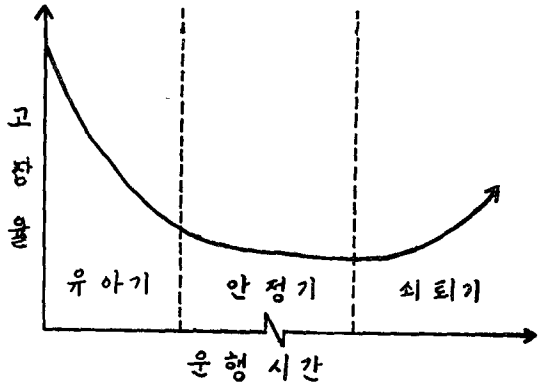

그림 2. 시스템수명과고장율

처음 使用 수 개월 동안은 相對的으로 故障率이 높은데 이 기간을 幼兒期라 하여 信賴性이 크게 問題가 되는 기간이다. 그대로 使用者에게 넘겨주고 集中的인 維持및 補修할 수도 있고 幼兒期를 生產工場(in-house)에서 거친 후 使用者에게 넘길 수도 있다.

5) 基本公式

① 信賴度를 나타내는 公式

平均故障간격(MTBF)

$$= \frac{T_1 + T_2 + \cdots\cdots + T_n}{n} \quad (1)$$

$$\text{平均修理時(MTTR)} = \frac{D_1 + D_2 + \cdots\cdots + D_n}{n} \quad (2)$$

$$\text{運行率} = \frac{\text{運行時間}}{\text{運行時間} + \text{故障時間}} \quad (3)$$

T_n : 各 運行時間

D_n : 各 故障時間

n : 回數

② 연결方式에 따른 시스템 信賴度

$$\text{直列系} : R_S = \prod_{i=1}^{n} r_i \quad (4)$$

$$\text{並列系} : R_P = 1 - \prod_{i=1}^{n} (1 - r_i) \quad (5)$$

R: 시스템 信賴度

r_i: 各 구성要素의 信賴度

③ 試驗및 運行에 의한 MTBF의 향상

$$T = T_0 \exp\left(\frac{\tau}{N_0 T_0}\right) \quad (6)$$

$$\Delta m = M_0 T_0 \left[\frac{1}{T_1} - \frac{1}{T_2}\right] \quad (7)$$

T, T_0, T_1, T_2 : MTBF 現在, 初期, 특정時點

τ: 現在까지의 運行時間

Δ: MTBF 향상을 위해 수정되는 故障의 수

N_0: 시스템에 들어있는 총 error의 수

M_0: N_0를 다 찾기 위한 故障의 수

3. 缺陷包容設計(Fault Tolerance Design)

1) 必要性

① Hardware의 部品이나 各 구성要素들의 故障率을 zero로 할 수 없다.

② 많은 경우 缺陷은 일시적이거나 복구할 수 있다.

③ Software의 경우 狀態의 數가 너무 많으므로 完全한 正確性을 기할 수 없다.

④ 入力되는 data가 完全하지 않다.

⑤ 部品의 質에만 依存하는 것은 비經濟的이다.

⑥ 시스템의 幼兒期나 약간 不安全한 狀態를 극복할 수 있다.

2) 方 法

① 가능한 모든 缺陷을 찾는다.
② 하나의 缺陷에 의한 障碍를 制限한다.
③ 가능한한 缺陷을 없었던 것으로 한다.
④ 重複裝置(redundancy)를 붙인다.
⑤ 주요부분을 固定시킨다.
⑥ 巨視的으로 시스템 監視를 한다.
⑦ 복구할 수 있는 준비를 한다.

4. 二重化

1) 二重化의 方式

信賴性 향상을 위한 重複裝置를 하는 方法에는 단순한 덧붙임, 二重化, 多重化, 多重化에 의한 分散등의 여러가지가 있으나, 우선 여기선 현재 널리 利用되고 있는 二重化에 對하여 설명하였다.

主制御裝置인 컴퓨터시스템의 二重化에는 그 運用方式에 따라, 完全히 분리되어 運用되고 서로간의 情報交換이 없어 故障時에도 시스템의 交替 이외는 對處가 可能하지 않은 別個시스템(Separate System), 서로 분리되어 있으나 적당한 方法에 의해서 作業을 分配하고 故障時에는 全作業을 한 컴퓨터가 담당하는 協同시스템(Cooperative System), 밀접하게 연결되어 있으나 두 컴퓨터간에 主從關係가 있어서 主業務와 시스템의 管掌은 主컴퓨터가 하고 從컴퓨터는 補助作業과 主컴퓨터의 故障을 對備하는 主從시스템(Master/Slave System), 밀접하게 연결되어 있고 두 컴퓨터 서로간에 data와 그 處理結果를 比較, 檢討하여 處理및 故障에 완벽을 期하는 並列시스템(Dual System)등을 들 수 있는데 다음에 설명할 GTK-500은 主從시스템이다.

2) 二重化의 長短點

二重化의 長短點은 우선 公式(4), (5)에서 기대할 수 있는 信賴度 향상이다. 信賴度 0.9인 두 개의 다른 시스템으로 한 시스템을 만들 때 二重化 시켰을 때와 그냥 했을 때를 비교해 보면

$$R_S = 0.9 \times 0.9 = 0.81$$

$$R_P = [1-(1-0.9)\ (1-0.9)]\ [1-(1-0.9))\ 1-0.9)] = 0.9801$$

다음으로 시스템 전체적인 면에서나 software에서 缺陷包容設計가 可能하다. 一般的으로 缺陷包容設計에는 重複裝置가 필요한데 특히 software에서 高級技術을 구사하려면 컴퓨터의 二重化가 꼭 필요하다. 또한 시스템에 필요한 部品들의 仕樣範圍를 넓게 잡을 수 있어 部品의 選定및 購買가 용이하다는 것도 長點으로 들 수 있다.

短點으로는 첫째 二重化에 의해서 價格이 비싸질 可能性이 있고 둘째 연결접속(interconnection)이 늘어남으로 인한 信賴度의 低下(統計的으로 電子裝置에 있어서의 많은 故障은 연결접속 부분에서 일어나는 것으로 되어 있다). 세째 二重化에 의한 制御方法이나 시스템 設計의 어려움등을 들 수 있다.

5. GTK—500 EPABX(電子式 構內交換機)

缺陷包容設計의 한 例로서 GTK-500 電子式

構內 交換機를 설명하겠다. GTK-500은 時分割 方式의 電子交換機로서 회선용량이 內線 500회선, 시내중계 100회선이며 또한 최종용량이 1,500회선까지 확장이 가능하도록 設計하였다. 通話路, 컴퓨터, 전원장치의 二重化와 缺陷包容設計를 하여 信賴性 향상을 도모하였다.

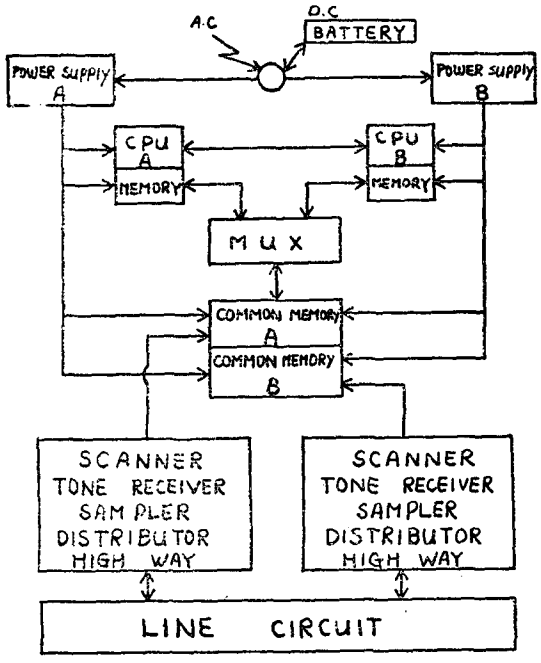

그림 3. System Block Diagram

1) 시스템 構成

信賴性을 위주로 볼 때 GTK-500은 그림 3에서와 같다. 加入者는 通話路에 並列로 連結되어 있으나 컴퓨터의 配定에 따라 Common memory를 通하여 어느 한쪽으로 接續된다. speech path test, common memory test, status test 등으로 어느 한쪽의 故障이 發見되면 故障난 通話路는 시스템에서 分離된다. 電源 역시 二重化로 構成되어 있는데 通話路의 경우는 並列로 連結되고 컴퓨터의 경우는 각각 따로 連結된다.

컴퓨터는 그림 4에서와 같이 두개의 CPU와 그 각각에 instruction들이 들어 있는 ROM (Read Only Memory), 使用者 data나 重要한 시스템 data를 保管하는 protected RAM(Random Access Memory), 變하는 data를 保管하는 RAM으로 이루어지는데 protected RAM은 key에 의하여 set/reset되며 사람에 의해서만 變更이 可能한 部分과 사람과 시스템에 의해서 flag를 set/reset함으로써 變更이 可能한 部分으로 나누어진다. 두 컴퓨터간에는 interrupt나 status test등을 통하여 서로 情報를 交換한다. 각 common memory에는 그 所屬 group의 誤動作與否를 나타내는 status가 있어 컴퓨터는 주기적으로 이를 읽어냄으로써 시스템의 狀態를 把握한다. 또 交換臺를 겸하는 supervision console이 있어서 이것을 통하여 綜合的인 시스템의狀態를 알 수 있고 對處할 수 있는 조작이 可能하다.

2) Software의 缺陷包容設計

GTK-500은 그림 5에서와 같이 主從 시스템으로 되어 있는데, 主컴퓨터는 전시스템의 監視, 作業分配, 通話處理및 自體診斷등을 遂行하는데 診斷結果 缺陷이 발견되면 緊急處理에 보내어지고, 거기서 缺陷의 輕重을 分析하여 原狀복귀, 統計, 경보및 분리등의 처리를 하며 가장 重症의 경우 control transfer를 한다. 從컴퓨터는 主컴퓨터로 부터 必要한 情報를 받아 data base를 준비하거나 自體診斷을 하고 主컴퓨터의

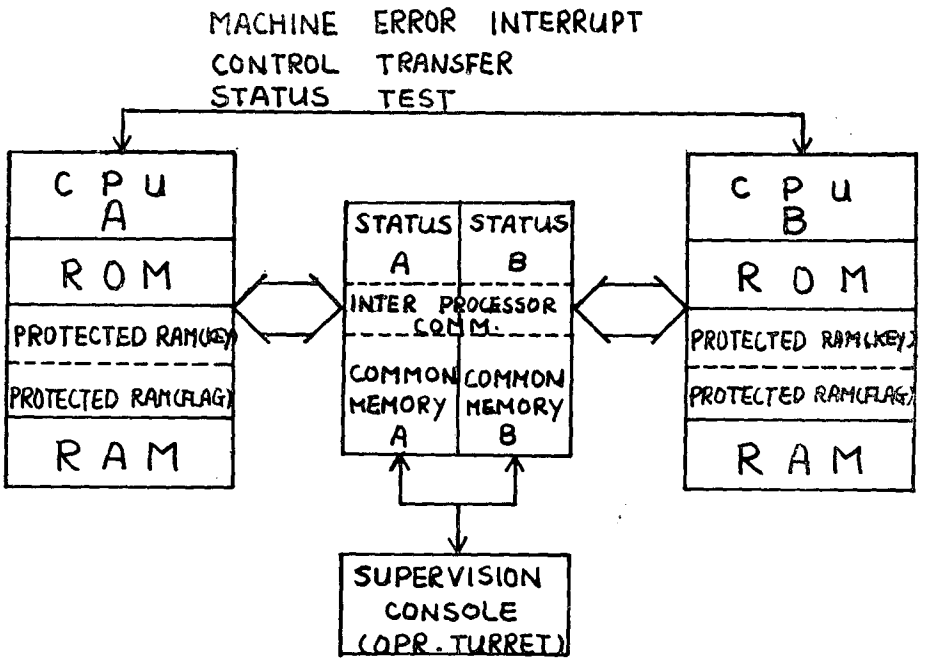

그림 4. Computer System Block Diagram

狀態를 巨視的으로 감시(supervising)한다. 그리하여 主컴퓨터의 고장을 발견하면—interrupt, 主컴퓨터의 control transfer, 巨視的 감시의 결과—즉시 restart하여 主컴퓨터의 任務를 遂行하게 된다.

6. 結 論

좋은 시스템이 되려면 그 機能도 좋아야 하지만 信賴性이 있어야 한다. 信賴性은 단순히 正確性보다는 缺陷包容性을 뜻한다. 이 缺陷包容

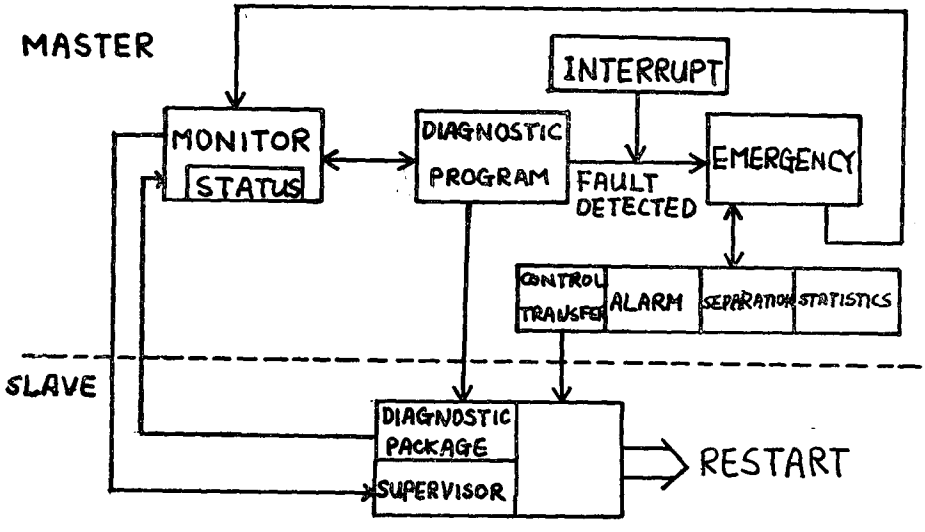

그림 5. Software Function Diagram

性은 software에 의해서 많이 좌우되는데 컴퓨터에 의해서 統制및 제어되는 시스템에 있어서는 특히 設計時에 이것을 많이 고려하여야 한다 缺陷包容設計는 여러가지 理由로 꼭 필요하며 그 一般的인 方法은 缺陷이 일어났을 때 어떻게 하면 복구시킬 수 있는가 이다. 二重化는 缺陷包容設計의 좋은 한 方法이며 이 二重化와 software에서 缺陷包容設計를 시도한 GTK-500은 좋은 信賴性이 기대된다. 모든 설명에 있어서 시스템 전반에 너무 광범위하게 언급한 느낌이 들며 실제로 이 때문에 說明에 隘路가 많았다. 다음엔 컴퓨터시스템이나 software의 缺陷包容設計에만 局限해서 說明할 기회를 기대해 본다.

參考文獻

1. GTK-500 시스템
2. "Reliable Hardware/Software Architecture", William A. Wulf, SOFTWARE ENGINEERING, June 1975, Volume SE-1, No.2
3. "System Structure for Software Fault Tolerance", Brian Randell, SOFTWARE ENGINEERING, June 1975, Volume SE-1, No.2
4. "Reliability Experience with Chi/OS", William C. Lynch, SOFTWARE ENGINEERING, June 1975, Volume SE-1, No.2
5. "A Theory of Software Reliability and Its Application", John D. Musa, SOFTWARE ENGINEERING, Sept. 1975, Volume SE-1, No.3
6. "Reliability" Signetics integrated circuits 1976.
7. "Processor Management", Sturt E. Madnick, Operating Systems, McGraw-Hill Book Company, 1974.
8. "AXE 10 SOFTWARE SECURITY", Kjell Svrme, INTERNATIONAL SWITCHING SYMPOSIUM, 1976.