分散型を含む計算機システムの信頼性評価手法 ISCP/S-R：Reliability

北嶋 弘行
（日立製作所 システム開発研究所）

1. 緒言

システム建設の初期段階でユーザ要求を満す経済的なシステム構成を導くこと（以下，これをシステム構成設計と呼ぶ）は，システム建設の要である。しかし，顧客ニーズ，およびソフトウエア・ハードウエア機種の多様化に伴い，計算機システムは全体として大規模・複雑化しつつあり，システム構成設計は困難かつ長期化しつつある。分散システム化はこの傾向に拍車をかけている。

従来からシステム建設の初期段階に適した性能予測ツールはいくつか提案されてきている。しかし，分散システムのように大規模なシステムの構成設計にこれらを適用しようとすると，次の問題がある。

(1) 性能だけでなく同時に信頼性評価が必要なことが多い。

(2) 従来の評価手法を大規模システムに踏襲したのでは，計算時間の長大化，モデル構築の工数が大などの問題がある。

(3) (1)の評価結果，目標水準に比べ過不足を生じた場合，従来のような試行錯誤的な方法では，適正なシステム構成案を得るまでに長時間を要し，しかも，これに至る保証もない。

上記の問題に対処するために，報告者らは分散型を含む計算機システム構成を，性能だけでなく信頼性の面から評価し，さらに改善案を自動作成する技法ISCP/S（Integrated tools for System Configuration Planning/Synthesis）を提案し，このTSS対話型ツール化を行ってきている。

性能面の支援機能は既に報告[1]した。そこで，本報告の以下では，先ず第2章でISCP/Sの枠組を示した後に，信頼性評価のためのサブシステム（ISCP/S-R：RはReliabilityの略）に限って述べる。

2. ISCP/Sの枠組

ISCP/Sは構成設計の各ステップを次の機能で共通的に支援する。すなわち，各設計段階で得られる処理要求仕様とシステム構成第一次案を入力とし，

(1) 先ず，この性能指標（各処理要求の平均応答時間，各資源の利用率）と信頼性指標（各処理要求のアベイラビリティや障害回復時間，および部分故障時のシステム処理能力）を見積る。

(2) 上記の結果，目標水準に対し過不足がある場合，この改善案を負荷配分と機器構成の観点から自動作成し，ガイダンスする。

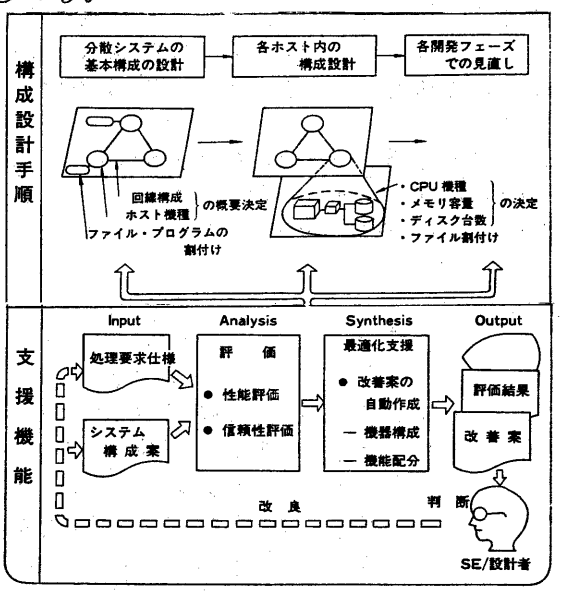


図1 ISCP/Sの枠組

3. 信頼性評価機能の設定

計算機システムの構成設計においては，エンドユーザにとっての信頼性を評価する必要がある。そのため，評価尺度としては，各処理要求タイプにとっての下記を用いることとした。

(a) 定常アベイラビリティ
(b) 平均回復時間
(c) 部分故障時の性能指標（スループットおよび平均応答時間）

上記(a),(b)は，システムに共通する評価尺度である。しかし，計算機システム（特に分散型）の場合，次の2点が課題となる。

(i) 分散型の場合を含む定常アベイラビリティ評価機能

特に分散型の場合，多数の構成要素が相互に関連している。また，1つの処理要求が異なった計算機に配置された複数の機能を要求することが多い。さらに，1つの機能のコピーが分散配置され，相互に代行しあえることが多い。加えて，処理要求は網内に複数の代替パスをもちうる。

これらを信頼性評価モデルの観点からみると，分散システムは極めて複雑な冗長系としてモデル化され，この解析手法が問題となる。

(ii) 時間と論理を含む場合の信頼性評価機能

ソフトウェアによる柔軟な動作メカニズムは，上記(i)のような冗長モデルでは表現できないことがある。この場合，AND，ORなどの同期の論理を自在に含み得るモデル化手法とその解析方法が必要である。さらに，障害回復には，上記以外に時間遅れを伴う。以上より，同期の論理と時間遅れをともに扱い得るモデル化方法とこの解析手法が必要と考える。

一方，上記(c)は，やや計算機システムに固有の尺度であり，次が課題となる。

(iii) 部分障害時における性能劣化の評価機能

分散システムの狙いの1つは，部分障害時にもあるレベル以上のサービスを提供可能とすることである。この評価には，部分故障時における処理フローパターンならびにその性能指標値を予測する手法が必要である。

以下の各章で，上記の課題に対する提案内容を，(iii)(i)(ii)の順に述べる。

4. 部分障害時の性能劣化・評価モデル

部分故障（同時に複数のサブシステムが故障しても良い）時における各処理要求の性能指標を評価するモデル，ならびに改善策をガイダンスするモデルを提案する。

4.1 フローパターン予測モデル

先ず，正常時におけるフローパターンの予測方法を述べる。計算機網の流量制御方法には，例えば次の方法がある。

① 各処理要求の通る経路が予め固定のもの（permanent circuit法）
② 各処理要求に対して何通りかの経路候補を設け，この間で順位づけしておき，出来るだけ順位が高く，かつ混雑度が閾値以下の経路に配分する方法。
③ 各処理要求に対する経路候補集合のうち，セッション設定時に最適な経路に割り当てる方法（virtual circuit法など）
④ 各パケットを各時点で最適な経路に配分する方法（適応制御法など）。

このような各処理要求の経路選択行動の結果に生じる網全体のフローパターンを正しく予測することは容易でない。シミュレーションも考えられるが計算時間の点で問題である。そこで，この近似的な予測手法であるIA（Incremental Assignment）法[3]を採ることとした。IA法適用の準備として，全ノード対にあふれ分用のダミー枝を増設し，その所要時間を大きな値とする。IA法は次の2段階からなる。

〔フェーズ1〕 (i) フローが全く流れていない状態から出発する。

(ii) 各処理要求tについて，この経路候補集合L_tの中から現フローパターンにおける最短経路l_t^*を選び，tの処理要求量λ_tのε%を割り当てる。（ここで，流量制御が上記②の場合は，L_t中の冗長な経路から調べ，混雑度が閾値以下のものにλ_tのε%を配分する。）

(iii) (ii)の操作を$100/\varepsilon$回繰返す。

〔フェーズ2〕

(i) 各処理要求tについて，tが流れる各経路lからtの流量f_{lt}のε%を除く。

(ii) 上記(i)の結果に得られたフローパターンにおいて，各処理要求tについてL_tの最短経路l_t^*を選び，これに$\varepsilon \cdot f_{lt}/100$を配分する。

(iii) 上記(i)(ii)を，いずれの処理要求に関してもフローパターンの変化が起らなくなるまで反復する。

上記のIA法の結果，各処理要求tのスループットf_tや平均応答時間T_t，および計算機網の各構成要素の利用率などの性能指標値が予測できる。

以上は，正常時におけるフローパターンの予測手法である。部分故障の場合には，故障している節点や枝を除去して上記のIA法を実施することにより，同様の性能指標値を計算できる。

4.2 フローパターンの改善モデル

上記は正常時および部分障害時のフローパターン予測モデルであった。予測の結果，目標値を達成できないことがある。この場合，目標値に出来るだけ近づける流量配分案を作成することが，資源活用のうえで大切である。そこで，各処理要求tに対するスループット要求量λ_tを出来るだけ満す流量配分を決定するモデルを提案した。

従来モデル[4]は系全体としてのスループット最大化を目的としていた。一方，提案モデルは，各処理要求のスループットや平均応答時間の要求を出来るだけ満すことを目的にしている。したがって，提案モデルによれば，よりきめ細い対策ができると期待できる。

〔フローパターンの改善モデル〕 ---(1)

（目的関数）各処理要求tのスループットを各々の目標値λ_tに近づける：

$$\sum_t w_t \left(1 - f_t/\lambda_t\right) \longrightarrow \min.$$

（制約条件）

応答時間制約： $R_t \leq R_t^*$, $t = 1, \cdots, T$

容量制約 ： $\sum_t x_{it} \triangleq \tilde{x}_i \leq C_i$, $i = 1, \cdots, I$

連続性 ： $\sum_{i \in \delta^+(v_k)} x_{it} - \sum_{i \in \delta^-(v_k)} x_{it} = \begin{cases} f_t & (v_k = O_t) \\ 0 & (v_k \neq O_t, D_t) \\ -f_t & (v_k = D_t) \end{cases}$

$t = 1, \cdots, T$

フロー要求 : $0 \le f_t \le \lambda_t$, $t = 1, \cdots, T$

ここで, $\delta^+(v_k), \delta^-(v_k)$: 節点から出る(+)および入る(−)枝の集合
O_t, D_t : 処理要求 t の出発側および到着側の節点
λ_t : 処理要求 t のスループット要求値
f_t : 処理要求 t のうち網に流せた流量
I : 枝の総数
T : 処理要求の総数
R_t, R_t^* : 処理要求 t に対する平均応答時間とその許容値
x_{it} : 枝 i を流れる処理要求 t の流量
C_i : 枝 i の許容流量. 該枝の故障時は $C_i = 0$ とおく.
w_t : 処理要求 t の1%を処理不能なことにより, $w_t/100$ の損失があることを示す.

上記の多種流問題にIA法を適用可能とするためには, これを, 容量制約なし, 応答時間制約なしで, 連続微分可能な凸関数の目的関数をもつ計画問題に変換する必要がある. そこで, 全ノード対にダミー枝 $i = I+1, \cdots, I+M$ (M: ノード対の総数) を増設し, この修正された網のうえで次の計画問題を定義する.

〔変形したフローパターンの改善モデル〕 --- (2)

(目的関数) $\Sigma_t \left[\Sigma_{i=I+1, I+M} (w_t x_{it}/\lambda_t) + \pi_t(g, R_t) \right] + \Sigma_{i=1, I} \psi(p, \tilde{x}_i) \to min.$

(制約条件) 連続性, フロー要求 : (1)式と同じ.

ここで,

$\psi(p, \tilde{x}_i)$: 容量制約の罰金関数. p (>0) はパラメータ, $\partial\psi/\partial\tilde{x}_i$ は $\tilde{x}_i$ に関して単調増加, かつ,

$$\lim_{p\to\infty} \psi(p, \tilde{x}_i) = \lim_{p\to\infty} \frac{\partial}{\partial \tilde{x}_i} \psi(p, \tilde{x}_i) = \begin{cases} 0 & \tilde{x}_i \le C \\ \infty & \tilde{x}_i > C \end{cases}$$

なる関数である。例えば, $\frac{1}{p} e^{p(\tilde{x}_i - C)}$ が該当する。

$\pi_t(g, R_t)$: 応答時間制約の罰金関数. g (>0) はパラメータ, $[\partial \pi_t(g, R_t)/\partial R_t] \cdot [\partial R_t/\partial \tilde{x}_i]$ は $\tilde{x}_i$ に関して単調増加, かつ

$$\lim \pi_t(g, R_t) = \lim_{g\to\infty} \frac{\partial \pi_t}{\partial R_t} \frac{\partial R_t}{\partial \tilde{x}_i} = \begin{cases} 0 & R_t \le R_t^* \\ \infty & R_t > R_t^* \end{cases}$$

なる関数である。例えば, $\frac{1}{g} e^{g(R_t - R_t^*)}$ が該当する。

なお, 性能目標の間で, 目標達成の優先度をつけたい場合には, (2)式において, w_t/λ_t や $\partial\psi/\partial\tilde{x}_i$ および $[\partial \pi_t/\partial R_t] \cdot [\partial R_t/\partial \tilde{x}_i]$ の値の大小で表現すればよい.

(2)式に対してIA法を次のように適用する.

〔フェーズ1〕

(i) フローが全く流れていない状態から出発する

(ii) 各処理要求 t について, L_t の各経路に関し, その流量を $\lambda_t \cdot \varepsilon/100$ だけ増加させた場合の目的関数の増分を計算し, L_t の全経路に関して増分が最小な経路を l_t^*, 増分を Δ_t^* とする. 次に, 全ての処理要求の Δ_t^* を比較し, Δ_t^* が最小となる処理要求を t' とする. 続いて, t' の流量 $\lambda_{t'} \cdot \varepsilon/100$ を経路 $l_{t'}^*$ に配分する. 以上の結果から, フローパターンを修正する.

(iii) (ii)の操作を, 全ての処理要求量を配分し終るまで行う.

〔フェーズ2〕

(i) 各処理要求 t について，t が流れる各経路から t の流量の ε% を取り除く。

(ii) 全ての処理要求の流量が ε% 除去された状態で，全処理要求 t について，$\angle_t$ の各々について現フローパターンの流量を $\lambda_t \cdot \varepsilon/100$ だけ増加させた場合の目的関数値の増分を計算し，$\angle_t$ の全経路に関して増分が最小な経路を l_t^*，その増分を Δ_t^* とする。次に，全ての処理要求の Δ_t^* を比較し，$\Delta_{t_0}^*$ が最小となる処理要求 t_0 の流量 $\lambda_{t_0} \cdot \varepsilon/100$ を経路 $l_{t_0}^*$ に配分する。

(iii) (ii)の操作を，いずれの t に関しても配分が完了するまで行う。

(iv) (i)～(iii)の操作を目的関数値の改良が起らなくなるまで続ける。

5. 確率論的な信頼性評価モデル

5.1 分散型の場合を含む定常アベイラビリティ評価モデル

分散システムの場合も含め，計算機システムの構成を，図3に例示するような確率的ネットワークで表す。分散型の場合，節点は計算機，通信プロセッサなどに，枝はこれらを結ぶ回線に，それぞれ対応する。また，計算機内の場合，節点は中央処理装置，主メモリ，外部記憶装置，入出力制御装置などに対応し，枝はこれらの結合関係を表す。さらに，節点や枝には，対応するシステム構成要素の定常アベイラビリティ値が付与されているものとする。

一方，任意の処理要求 t を図2に例示するごとくモデル化する。すなわち，機能（プログラムやファイル）あるいは節点の系列で表わされるものとする。さらに，各処理要求 t の遂行に関して次を仮定する。

〔仮定1〕 各処理要求 t が遂行されるための条件は，t に含まれる全ての連続する機能や節点の対が遂行されることである。

〔仮定2〕 複数コピーをもつある機能について，該機能に対する処理が複数コピー間の即時更新化を要する場合以外は，1コピーが可用であれば該機能は遂行できる。

分散型のように複雑なシステムの場合，これを直並列グラフに変換することはむつかしく，カットやパスの探索に依らざるを得ない[57]。4.1節の冒頭に述べたように，計算機網で各処理要求の通り得るパスは限定されることがある点に着目し，ここではパス探索法の方をとることとした。各処理要求のパス探索は上記の2仮定を組み込んだバックトラッキングで実現している。

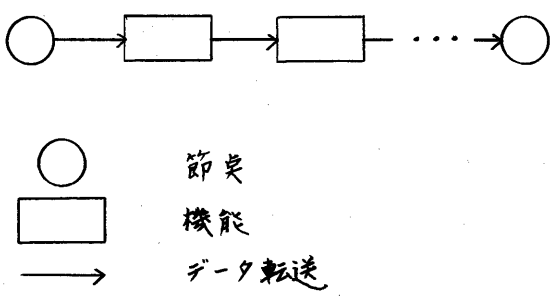


図2. 処理要求の例

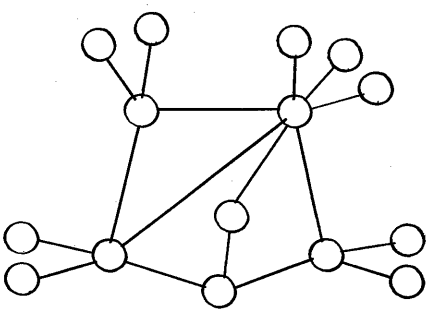


図3. システム構成を表す確率的ネットワークの例

5.2 時間と論理をともに含む系に対する信頼性評価モデル

処理間の同期の論理が入り混ったシステムでは，5.1の手法によってもモデル化が難しいことがある．この場合，論理をより明示的に取り込める手法が望ましい．さらに，障害回復のモデル化においては，上記の他に時間遅れの要素を取り込めることが望ましい．そこで，時間と論理を明示的に扱い得る確率的ネットワーク手法の導入を試みた．

この手法の特徴は次の2点に要約できる．

① GERT (Graphical Evaluation & Review Technique)[6]図で信頼性モデルを記述し，

② さらに，枝の時間値が確定的である場合，これを解析的に扱う．

(1) GERT図による信頼性モデル

GERT図の枝は処理を，節点は枝が実現するための論理的条件を表す．さらに，枝は処理の成功確率や所要時間のパラメータをもつ．(図4参照)

GERT図の簡単な適用例を図5に示す．処理要求aは，通常は，分散側計算機でファイルa，bをともに用いて実行される．しかし，これらの障害時には中央側計算機で同ファイルのバックアップ・コピーを用いて実行される．図5は，処理要求aにとっての定常(アン)アベイラビリティと平均回復時間の評価モデルを表したものである．この図で，各枝iは次のパラメータをもつ．

(i) 成功確率 … 該要素iの定常アベイラビリティ a_i

(ii) 失敗確率 … 該要素iの単位時間当りの平均故障回数(すなわち，故障生起率)．これは，$\lambda_i a_i$に等しい(λ_i：iの故障率)．

(iii) 所要時間 … もし，障害回復に伴う所要時間ならば修理時間を，その他の遅れ時間ならばその所要時間を用いる．

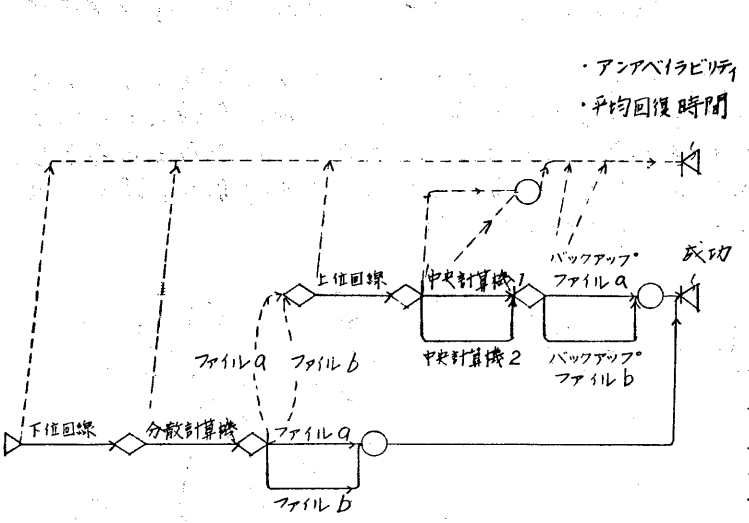
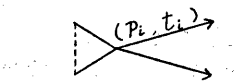


図5 GERT図による信頼性モデルの例

枝 = 処理



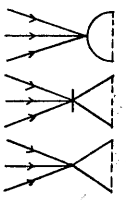
p_i：処理iの分岐確率

(例) アベイラビリティ
故障生起率(回/月)

t_i：処理iの所要時間

(例) 修理時間

節点 = 処理開始の論理的条件



AND

Exclusive-OR

Inclusive-OR

図4 GERT図の要素

(2) 枝時間が確定的な場合の解析方法

GERT図に対する解析的手段の適用範囲は，従来，節点の論理がExclusive-ORのみからなるGERT図に変形できる場合に限られ，この他はシミュレーションが行われてきた．しかし，シミュレーションは計算時間が大きいという欠点がある．特に，信頼性評価の場合のように低い分岐確率をとる問題に対しては，大きな問題である．そこで，枝の所要時間が確定的と仮定できる場合に限り，GERT図を解析可能化する方法を提案した[3]．この方法を下記する．

(a) 付点つきGERT図 ： 元のGERT図に次の修正を加える．

(i) 遅れ節点 ； 時間＞0なるすべての枝を，遅れ節点と時間＝0の枝で置き換える． 遅れ節点には元の枝の時間パラメータと，これを解消するのに必要な残り時間の2パラメータを与える．(図6)

(ii) 付点 ； GERT図の節点のうち実現しているものに付点をつける．ここで遅れ節点の場合，新たに付点がついた時点でその残り時間＝時間パラメータに設定され，時間が経過し残り時間＝0となった時点で初めて実現するものとする．

(b) 付点の移動規則 ： 付点つきGERT図における付点の移動規則を図7の例で示す．節点n_1, n_2は成立しておりn_3は未成立である．n_3の論理別にn_3の実現確率（遷移確率と呼ぶ）を表1に纏める．

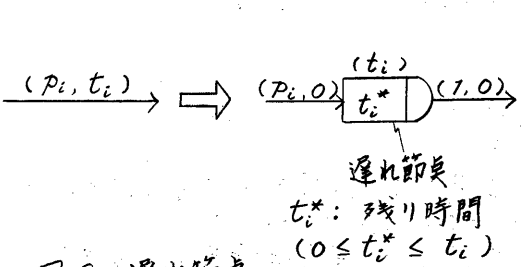


図6 遅れ節点

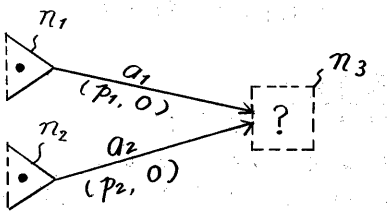


図7 付点つきGERT図の例

表1 付点の移動規則の例
（図7に対応）

n_3の論理	実現枝集合	遷移確率
Exclusive - OR	(a_1)	$p_1 - p_1p_2$
	(a_2)	$p_2 - p_1p_2$
Inclusive - OR	(a_1, a_2)	p_1p_2
	(a_1)	$p_1 - p_1p_2$
	(a_2)	$p_2 - p_1p_2$
AND	(a_1, a_2)	p_1p_2

(c) 状態遷移図 ： 付点つきGERT図における状態Sを次のように定義する．

$S = (\delta_1, \cdots, \delta_i, \cdots, \delta_m, t_1^*, \cdots, t_j^*, \cdots, t_n^*)$

ここで，δ_i ：節点iが付点をもっているか否か．

t_j^* ：遅れ節点jの残り時間．

付点付きGERT図を(b)の規則で動作させることにより，状態遷移図を得る．状態遷移図は状態を節点で，遷移を有向枝で表す．枝は遷移確率と遷移時間をパラメータとする．遷移確率の求め方は(b)で述べた．遷移時間の求め方は次のごとくである．付点つきGERT図で新たに成立可能な節点がなくても，遅れ節点の残り時間を一律にある値Δtだけ減少することで，付点が移動可能となることがある．この場合，遷移時間＝Δtとする．

以上のようにして得た状態遷移図はシグナルフロー理論などで解析可能である．

6. 結　言

分散型を含む計算機システム構成の最適設計をＴＳＳで支援する技法ＩＳＣＰ/Ｓにおける信頼性評価機能を述べた。計算機システム構成設計における信頼性評価に関して３つの課題を挙げ，この解決策を提案した。課題の第１として，部分障害時でのシステム全体の性能劣化評価ならびに最適負荷配分の手法をとりあげた。このため，先ず，各種・流量制御方式の表現方法としてＩＡ法を採用した。次に，最適負荷配分については，総スループットの最大化を目的とする従来手法を発展させ，各処理要求タイプごとに平均応答時間要求をまもり，スループットを目標値に近づけることが出来るモデルと，このＩＡ法を用いた近似計算手法を提案した。第２の課題として，分散型を含むシステムに対する定常アベイラビリティの評価手法をとりあげた。従来の確率的グラフ上でのノード対間パス探索法を発展させ，各処理要求が複数の機能を要求し，しかも各機能は複数コピーを持ち得る場合の，パス探索法を提案した。第３の課題として，時間と論理を含む系に対する確率的な信頼性評価手法をとりあげた。上記の場合がＧＥＲＴ図で表現できることを示し，さらに，枝時間が確定的な場合，付点を導入し状態遷移図化することにより，ＧＥＲＴ図を解析可能化する手法を示した。

本報告内容により，品質向上や工数低減が期待できる。今後の課題として，ソフトウェア信頼性に関するフィールドデータの蓄積が必要である。

7. 参考文献

1) 北嶋 他：計算機システムの構成設計支援技法ISCP/S：Synthesis，情処学会・本部発会資料 17-6 (1982)

2) 北嶋 他：性能および信頼性面からの計算機システム構成・最適設計支援技法 ISCP/S，情処学会第25回大会，pp. 379-380 (1982)

3) 北嶋 他：動的な事故波及過程に対する確率論的な安全性評価方法の提案，電気学会論文誌C，(1981-10)

4) 後藤 ：通信網設計へのＩＡ法の応用，オペレーションズ・リサーチ，pp. 711-719 (1977)

5) 猪瀬 他：コンピュータシステムの高信頼化，情処学会 (1977)

6) 土田 ：GERT入門(1)～(3)，オペレーションズ・リサーチ (1974)