(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5280587号
(P5280587)
(45)発行日　平成25年9月4日(2013.9.4)　(24)登録日　平成25年5月31日(2013.5.31)

(51)Int.Cl.　F I
G06F 11/30 (2006.01)　G06F 11/30　A

請求項の数 22　(全 77 頁)

(21)出願番号　特願2012-546760(P2012-546760)
(86)(22)出願日　平成23年11月14日(2011.11.14)
(86)国際出願番号　PCT/JP2011/076219
(87)国際公開番号　WO2012/073686
(87)国際公開日　平成24年6月7日(2012.6.7)
審査請求日　平成25年3月19日(2013.3.19)
(31)優先権主張番号　特願2010-267461(P2010-267461)
(32)優先日　平成22年11月30日(2010.11.30)
(33)優先権主張国　日本国(JP)
(31)優先権主張番号　特願2011-177322(P2011-177322)
(32)優先日　平成23年8月12日(2011.8.12)
(33)優先権主張国　日本国(JP)

早期審査対象出願

(73)特許権者　503360115
独立行政法人科学技術振興機構
埼玉県川口市本町四丁目１番８号
(74)代理人　110000338
特許業務法人原謙三国際特許事務所
(72)発明者　横手　靖彦
日本国東京都港区虎ノ門３－１４－１－２００４
(72)発明者　所　真理雄
日本国東京都大田区田園調布本町５－１５
(72)発明者　山本　修一郎
日本国愛知県名古屋市港区千鳥２－７－２０－３０１

最終頁に続く

(54)【発明の名称】ディペンダビリティ維持システム、変化対応サイクル実行装置、障害対応サイクル実行装置、ディペンダビリティ維持システムの制御方法、制御プログラムおよびそれを記録したコンピュータ

(57)【特許請求の範囲】
【請求項１】
対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムであって、
上記対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルを実行する変化対応サイクル実行装置と、
上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルを実行する障害対応サイクル実行装置と、
上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの変更要求を送信する変更要求送信手段と、を備えることを特徴とするディペンダビリティ維持システム。
【請求項２】
上記ディペンダビリティ記述データを格納するディペンダビリティ記述格納部を備え、
上記変化対応サイクル実行装置および上記障害対応サイクル実行装置が、上記ディペンダビリティ記述格納部に格納されている上記ディペンダビリティ記述データを共有しながら、上記変化対応サイクルおよび上記障害対応サイクルを一方ずつまたは両方同時に実行

することを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項３】
上記障害対応サイクル実行装置が上記障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、当該対象システムを再構成する再構成手段を備えることを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項４】
上記ディペンダビリティ記述データには上記再構成の手順が記載されており、
上記再構成手段は、上記ディペンダビリティ記述データに記載されている再構成の上記手順に従って、当該対象システムを再構成することを特徴とする請求項３に記載のディペンダビリティ維持システム。
【請求項５】
上記ディペンダビリティ記述データの変更要求を受信したとき、上記変化対応サイクル実行装置が上記ディペンダビリティ記述データを変更することを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項６】
上記ディペンダビリティ記述データを上記ディペンダビリティ記述格納部から取得し、上記対象システムのディペンダビリティの価値を定量的に示す評価値を求めるディペンダビリティ値決定手段を備え、
上記ディペンダビリティ値決定手段が、上記評価値を、上記変化対応サイクルまたは上記障害対応サイクルにおいて決定することを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項７】
上記ディペンダビリティ記述データは、互いに関連づけられたゴールノードとモニタノードとの組を規定するデータであり、
上記ゴールノードは、要求・仕様がゴール形式にて記述されたノードであり、
上記モニタノードは、上記ゴールノードに記述されたゴールが充足されていることを表明するエビデンスであるとともに、上記対象システムの監視点に対応付けられており、
上記ディペンダビリティ値決定手段は、対応付けられた上記監視点から取得されたデータがモニタノードに関連づけられた変動許容範囲に基づいて、上記評価値を計算することを特徴とする請求項６に記載のディペンダビリティ維持システム。
【請求項８】
上記ディペンダビリティ値決定手段は、上記対象システムをモニタして取得されたモニタ値が変動許容範囲に対して良好であったエビデンスに基づいて、上記評価値を計算することを特徴とする請求項７に記載のディペンダビリティ維持システム。
【請求項９】
上記障害対応サイクル実行装置が上記障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに含まれるスクリプトを実行するスクリプト処理手段を備え、
上記スクリプトは上記対象システムを変動許容範囲の状態に回復させるシナリオを含むことを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項１０】
上記対象システムの状態を監視し、必要な場合に対策を実行する障害監視制御部と、
上記ディペンダビリティ記述データを格納するディペンダビリティ記述格納部と、
上記障害監視制御部の動作を制御する障害監視制御用データを、上記ディペンダビリティ記述格納部から読み出した上記ディペンダビリティ記述データから生成して、上記障害監視制御部に供給するディペンダビリティ記述変換手段と、を備え、
上記障害監視制御用データに従って、上記障害監視制御部が動作することにより、上記対象システムの状態を監視し、必要な場合に対策を実行することを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項１１】

上記対象システムの状態を監視する１以上のモニタモジュールと、
必要な場合に上記対象システムに対して対策を実行する１以上のアクションモジュールと、
上記障害監視制御部の制御により、上記モニタモジュールの選択および動作の制御を行うモニタ管理部と、
上記障害監視制御部の制御により、上記アクションモジュールの選択および動作の制御を行うアクション管理部と、を含むことを特徴とする請求項１０に記載のディペンダビリティ維持システム。
【請求項１２】
他のディペンダビリティ維持システムとネットワークを介して接続されていることを特徴とする請求項１に記載のディペンダビリティ維持システム。
【請求項１３】
請求項１に記載のディペンダビリティ維持システムを構成する変化対応サイクル実行装置。
【請求項１４】
請求項１に記載のディペンダビリティ維持システムを構成する障害対応サイクル実行装置。
【請求項１５】
上記ディペンダビリティ記述データをディペンダビリティ記述格納部から読み出すとともに、
上記対象システムの状態を監視し、必要な場合に対策を実行する障害監視制御部の動作を制御するための障害監視制御用データを、読み出した上記ディペンダビリティ記述データから生成するディペンダビリティ記述変換手段を備えたことを特徴とする請求項１４に記載の障害対応サイクル実行装置。
【請求項１６】
上記障害監視制御部は、上記対象システムの状態を監視し、必要な場合に対策を実行するために使用するモジュールを、複数のモジュールから選択して制御可能であり、
上記ディペンダビリティ記述データは、１つの記述が、上記モジュールを特定するためのモジュール特定情報を値として設定可能な変数を少なくとも含む形式であり、
上記ディペンダビリティ記述変換手段は、あらかじめ設定された、上記モジュール特定情報と上記障害監視制御用データとの対応関係を示す情報に基づいて、上記ディペンダビリティ記述データに含まれる変換対象の記述を、該記述に含まれるモジュール特定情報に対応する障害監視制御用データに変換することを特徴とする請求項１５に記載の障害対応サイクル実行装置。
【請求項１７】
上記ディペンダビリティ記述データの１つの記述から変数部分を除いた固定部分に対応して、上記対応関係が設定されており、
上記ディペンダビリティ記述変換手段は、上記ディペンダビリティ記述データの記述に含まれる上記固定部分に対応する上記対応関係を参照して、該記述を対応する障害監視制御用データに変換することを特徴とする請求項１６に記載の障害対応サイクル実行装置。
【請求項１８】
対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムの制御方法であって、
上記ディペンダビリティ維持システムの変化対応サイクル実行装置が、上記対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルと、
上記ディペンダビリティ維持システムの障害対応サイクル実行装置が、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルと、

上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの変更要求を送信する変更要求送信ステップと、を含むことを特徴とするディペンダビリティ維持システムの制御方法。

【請求項１９】

上記ディペンダビリティ記述データをディペンダビリティ記述格納部から読み出す読み出しステップと、

上記対象システムの状態を監視し、必要な場合に対策を実行する障害監視制御部の動作を制御するための障害監視制御用データを、読み出した上記ディペンダビリティ記述データから生成する変換ステップと、を含むことを特徴とする請求項１８に記載のディペンダビリティ維持システムの制御方法。

【請求項２０】

請求項１から１２のいずれか１項に記載のディペンダビリティ維持システムの上記各手段としてコンピュータを機能させるための制御プログラム。

【請求項２１】

請求項１５から１７のいずれか１項に記載の障害対応サイクル実行装置の上記ディペンダビリティ記述変換手段としてコンピュータを機能させるための制御プログラム。

【請求項２２】

請求項２０または２１に記載の制御プログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、時間軸上で変化可能なディペンダビリティ要求に対して、ある時刻における当該要求を表現したディペンダビリティ記述データを備える対象システムにおけるディペンダビリティの価値を計測、評価することを可能にし、当該対象システムのディペンダビリティを維持する装置等に関するものである。

【背景技術】

【０００２】

昨今、銀行のオンラインシステムの停止、携帯電話やＩＰ電話の通信障害、各種商用サービスのセキュリティ障害、等の重要な社会インフラストラクチャを担うシステムやサービス停止が目立ってきており、我々の生活に影響を与えている。その原因は、ひとえにそれらの組み込み型コンピュータシステムを利用した商品やサービスにおいて、その規模や複雑度が劇的に増してきていることに帰することができる。その原因をさらに掘り下げて調べてみると、人為的なミスが原因であることがかなりのケースで存在している。

【０００３】

従来から、コンピュータシステムの信頼性、可用性、保守性、安全性、完全性、機密性に関しては、ディペンダビリティというコンピュータシステムの備えるべき性質として議論されてきている（非特許文献３）。組み込みシステム開発においては、最初に開発計画を立て、対象システムやサービスの機能要件および非機能要件を仕様としてきっちり書き出し、検証やテストを長期にわたって行い、デプロイする手法がとられてきた。しかし、上述のように、障害が日に日に件数を増してきている。ＣＭＭＩやＩＳＯ　２６２６２をはじめとする規格では、人為的エラーを減らす試みもなされている。しかし、これら既存の技術・規格では、オープン環境におけるシステムという特性に対する考慮が欠けている。

【０００４】

従来からの手法は、開発の開発時点における仕様が、コンピュータプログラムとして確実に実装され、かつその仕様が商品やサービスのデプロイ後も不変であるという仮定に基づいている。しかし、オープン環境では、開発の開発時点とデプロイ時点では環境が変化している。さらに、デプロイ後も環境は変化している。その結果、それらの変化への対応

が求められる。
【０００５】
そこで、独立行政法人科学技術振興機構は、CRESTプログラムにおいてDEOS（Dependable Embedded Operating Systems／Dependability Engineering for Open Systems）プロジェクト（http://www.crest-os.jst.go.jp/）を立ち上げ、組み込みシステム用のディペンダブル・オペレーティングシステムの研究開発を行っている。DEOSプロジェクトでは、オープン環境におけるディペンダビリティを、オープンシステムディペンダビリティとして次のように定義している。「現代の大規模ソフトウェアシステムは機能、構造、システム境界が時間的に変化し、これに起因する不完全さと不確実さを完全に排除することができず、未来に障害となりうる要因（開放系障害要因）を本質的に抱えている。オープンシステムディペンダビリティとは、それらの要因を顕在化する前にできる限り取り除き、また、顕在化した後に迅速かつ適切に対応し、影響を最小とするようにマネージし、利用者が期待する便益をできる限り安全にかつ継続的に提供し、社会への説明責任を全うし、およびそれらを継続的に行う能力を言う。」（非特許文献１参考）
また、従来、組み込みシステム開発においては（当該開発には限定されないが）、ステークホルダ群からの要求、及び当該要求に対する仕様書を作成し、それに基づいてシステム開発が行われている。具体的には、対象システムの機能要求、および非機能要求をまとめた仕様書群に従ってシステムは開発される。そして、運用中にシステムの一部を変更する場合には、仕様書群と当該システムの実装とが無矛盾に更新される。
【０００６】
当該仕様書群と当該システム実装の更新が無矛盾に行われる必要がある理由の一つとして、当該システムのディペンダビリティ（非特許文献１あるいは非特許文献３等を参照）は、いかに環境が変化しても維持しなければならないことがあげられる。そのため、当該仕様書に対応したディペンダビリティ記述データの更新と、それを実現するためのシステムを監視制御するモジュールの開発・追加が、常に当該仕様書群と無矛盾に行われることを保証しなければならない。
【０００７】
また、非特許文献２には、Safety Caseと呼ばれるシステムの安全性を示すドキュメントを、システムのライフサイクル（概念、開発、運用、保守、更新、廃棄など）を通じて、システムの変更に応じて更新することの必要性が記載されている。
【０００８】
Safety Caseは、イギリスなどで、原子力発電所など、高い安全性が求められるシステムを開発・運用する際に認証機関に提出が義務付けられるまでに普及している、システムの安全性の根拠（エビデンス）を示すための、構造化されたドキュメントである。自動車の機能安全規格であるISO 26262でも提出が義務付けられるなど世界的に普及しつつある。
【先行技術文献】
【特許文献】
【０００９】
【特許文献１】米国特許第７７５６７３５号明細書（２０１０年７月１３日）
【非特許文献】
【００１０】
【非特許文献１】Mario Tokoro, “Challenge to Open Systems Problems,” September 29, 2010. http://www.stanford.edu/class/ee380/Abstracts/100929.html [平成２３年７月７日検索]
【非特許文献２】Peter Bishop and Robin Bloomfield, “A Methodology for Safety Case Management,” Safety Critical System Symposium, 1998. http://www.adelard.com/papers/sss98web.pdf [平成２３年７月７日検索]
【非特許文献３】Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” IE

EE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, Jan.-Mar. 2004, doi:10.1109/TDSC.2004.2.
【非特許文献４】所眞理雄、JST-CREST「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム-DEOSプロジェクト-White Paper Version 2.0」、DEOS-FY201-WP-02J、2010/12/01
【非特許文献５】Jin Nakazawa, Yutaka Matsuno, and Hideyuki Tokuda, “Evaluating Degree of Systems' Dependability with Semi-Structured Assurance Case” Proceedings of the 13th European Workshop on Dependable Computing (EWDC 2011), 2011.
【非特許文献６】ＤＥＯＳ Ｃｏｒｅ Ｔｅａｍ, “Ｏｐｅｎ Ｓｙｓｔｅｍｓ Ｄｅｐｅｎｄａｂｉｌｉｔｙ Ｃｏｒｅ － Ｄｅｐｅｎｄａｂｉｌｉｔｙ Ｍｅｔｒｉｃｓ － オープン・システム・ディペンダビリティ・コア － ディペンダビリティメトリクス － ”, ［ｏｎｌｉｎｅ］, ２００９．０９．０４, 科学技術振興機構, ［２０１２年２月３日検索］, インターネット＜ＵＲＬ：ｈｔｔｐ：／／ｗｗｗ．ｃｒｅｓｔ－ｏｓ．ｊｓｔ．ｇｏ．ｊｐ／ｔｏｐｉｃｓ／ｄｅｏｓ２００９／ｍｅｔｒｉｃｓ．ｐｄｆ＞

【発明の概要】
【発明が解決しようとする課題】
【００１１】
システム規模の拡大や複雑度の増大は、当該システムやそれが提供するサービスに関する要求・仕様をも複雑にしている。そのため、すべての要求を完全にシステム開発前に抽出したり、全ての仕様を完全にシステム開発前に記述することが不可能（仕様の不完全性）になっている。仕様の不完全性は対応する実装も完全ではなく（実装の不完全性）、当該システムが提供するサービスの振る舞いも完全には把握することができない。その結果、「何をどこまで保証したら良いのか？」「保証可能なのか？」も分からない。また、これらの不完全性はステークホルダ間でのシステムやサービスに対する要求理解の違い（誤解）を招いたり、人為的なミスを誘導したりする。
【００１２】
さらに、先に述べた環境の変化に対応するための要求が新たに発生する。また、開発時点の要求にも修正が必要になる。「環境がどのように変化するか？」は事前には分かり得ないので、「現在のシステムの動作が、環境変化に確実に対応できるか？」もわからない。このような変化に対する不確実性はシステム動作予測を困難にし、システム障害に繋がる。
【００１３】
これらの不完全性や不確実性はシステム障害が発生した際に、その原因調査に時間がかかるようになり、またその適切な対応を困難にし、ステークホルダが自らの説明責任を果たすことも難しくなる。
【００１４】
このような状況下において、従来、非特許文献２においては、システム開発時点で想定される障害対応をSafety Caseとして記述し、ステークホルダ間で合意することで、障害発生時点でのステークホルダの説明責任の達成を可能にしている。しかし、環境の変化に起因するSafety Caseの変更には対応できない。
【００１５】
また、非特許文献３は、Faults-Errors-Failuresという変化で障害をとらえることで障害の発生そのものを抑えようとしている。しかし、前記不完全性や不確実性を扱うことはできない。
【００１６】
また、非特許文献５、６には、前記不完全性や不確実性を扱うための手法に関して記載されている。しかし、その記述は実行環境とは分離されており、また、記述をモデル化しディペンダビリティの価値を計測・評価することに関しては設計時のみにて可能であるに過ぎない。
【００１７】

一方、特許文献１はEnterprise Architectureにおける複雑性を数学的に制御する手法に関して述べている。複雑性が制御できればディペンダビリティの向上につながることが期待できるが、前記不完全性や不確実性までは扱っていない。

【００１８】

また、従来のシステム開発の手法では、複数のステークホルダ群からの要求は合意された仕様書にまとめられていなかったり、仕様書にまとめることが容易でないことに加えて、運用後にシステムの一部に変更があった場合に、仕様書とシステム実装の変更とを無矛盾に維持することができなかった。

【００１９】

例えば、上記Safety Caseでも、Safety Caseに記載されている、ステークホルダがSafety Case上で合意した障害対応内容と、実際の対象システムの障害対応やシステム監視制御部の開発・追加などとが無矛盾であることを維持する実用的なシステムは存在していない。このようなSafety Caseと対象システムとの無矛盾性は、ほぼ人手で行わなければならず、システムの更新と、Safety Caseの更新を無矛盾に行うことは、時間的なずれが生じるなど、問題点が多い。一般に、Safety Caseのような、認証機関などを含む、広範囲のステークホルダが理解可能な形式で記述されたドキュメントと、システムの実際の動作との間の無矛盾性を保証する実用的なシステムは、発明者が知る範囲においてはない。

【００２０】

本発明は、上記の問題点に鑑みてなされたものであり、その目的は、不完全性と不確実性が潜在的に存在するオープン環境において、対象システムのディペンダビリティの維持を支援することを可能にすると共に、対象システムの仕様書群と当該システムの実装を無矛盾に維持することを容易にするディペンダビリティ維持装置等を実現することにある。なお、本発明では、システムのディペンダビリティを非特許文献３などの定義に、複数のステークホルダ間での合意を含んだものに拡張する。

【課題を解決するための手段】

【００２１】

上記課題を解決するために、本発明に係るディペンダビリティ維持システムは、対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムであって、上記対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルを実行する変化対応サイクル実行装置と、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルを実行する障害対応サイクル実行装置と、上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの変更要求を送信する変更要求送信手段と、を備えることを特徴としている。

【００２２】

また、本発明に係るディペンダビリティ維持システムの制御方法は、対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムの制御方法であって、上記ディペンダビリティ維持システムの変化対応サイクル実行装置が、上記対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルと、上記ディペンダビリティ維持システムの障害対応サイクル実行装置が、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルと、上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの

変更要求を送信する変更要求送信ステップと、を含むことを特徴としている。

【００２３】

上記の構成によれば、障害対応サイクル実行装置において、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であると判断したとき、変化対応サイクル実行装置に対して、ディペンダビリティ記述データの変更の要求を送信することができる。

【００２４】

これにより、変化対応サイクル実行装置は、障害対応サイクル実行装置が送信したディペンダビリティ記述データの変更の要求を受信したとき、当該要求に応じて、上記ディペンダビリティ記述データを変更することが可能となる。

【００２５】

したがって、障害対応サイクル実行装置と、変化対応サイクル実行装置とが連携して、障害対応サイクルにおいて、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であれば、変化対応サイクルにおいて、ディペンダビリティ記述データを変更するという、一連のプロセスを円滑に実行することができる。よって、オープン環境において、対象システムのディペンダビリティを継続的に維持することが可能となるという効果を奏する。

【００２６】

なお、上記の変化対応サイクル実行装置および障害対応サイクル実行装置を含むディペンダビリティ維持システムは、コンピュータによって実現してもよく、この場合には、コンピュータを上記の各手段として動作させることにより上記ディペンダビリティ維持システムをコンピュータにて実現させるディペンダビリティ維持システムのプログラム、およびそれを記録したコンピュータ読み取り可能な記録媒体も、本発明の範疇に入る。

【発明の効果】

【００３５】

以上のように、本発明に係るディペンダビリティ維持システムは、対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルを実行する変化対応サイクル実行装置と、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルを実行する障害対応サイクル実行装置と、上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの変更要求を送信する変更要求送信手段と、を備える構成である。

【００３６】

また、本発明に係るディペンダビリティ維持システムの制御方法は、ディペンダビリティ維持システムの変化対応サイクル実行装置が、対象システムの開発時あるいは運用時における上記対象システムの更新時に、上記対象システムのディペンダビリティに関する要求・仕様を記述したディペンダビリティ記述データを上記対象システムに関する要求・仕様の変更に合わせて追加・修正する変化対応サイクルと、上記ディペンダビリティ維持システムの障害対応サイクル実行装置が、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルと、上記障害対応サイクル実行装置が障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクル実行装置に対して、上記ディペンダビリティ記述データの変更要求を送信する変更要求送信ステップと、を含む方法である。

【００３７】

それゆえ、障害対応サイクル実行装置と、変化対応サイクル実行装置とが連携して、障害対応サイクルにおいて、対象システムの障害発生または障害予兆を検知し、対象システ

ムの停止が不回避であれば、変化対応サイクルにおいて、ディペンダビリティ記述データを変更するという、一連のプロセスを円滑に実行することができる。よって、オープン環境において、対象システムのディペンダビリティを継続的に維持することが可能となるという効果を奏する。

【００４１】

本発明のさらに他の目的、特徴、および優れた点は、以下に示す記載によって十分に分かるであろう。また、本発明の利点は、添付図面を参照した次の説明で明白になるであろう。

【図面の簡単な説明】

【００４２】

【図１】本発明の一実施形態を示すものであり、ディペンダビリティ維持システムの構成の概略を示す機能ブロック図である。

【図２】図１に示したディペンダビリティ維持システムのワークスペースコンピュータおよびランタイムコンピュータのハードウェア構成を示すブロック図である。

【図３】図１に示したディペンダビリティ維持システムで用いるソフトウェアの構成例を示すブロック図である。

【図４】図１に示したディペンダビリティ維持システムで処理するアプリケーションの一例を示す説明図であり、（ａ）はアプリケーションが３層構造モデルをしていることを示し、（ｂ）は３層構造モデルを実装する際の一構成例を示す。

【図５】図１に示したディペンダビリティ維持システムで処理するアプリケーションのディペンダビリティ記述データの一例（正常系）を示す説明図である。

【図６】図１に示したディペンダビリティ維持システムで処理するアプリケーションのディペンダビリティ記述データの一例（リスクベース）を示す説明図である。

【図７】図６に示したディペンダビリティ記述データと、ステークホルダ要求変更に伴って変化したディペンダビリティ記述データとの差分を示すディペンダビリティ記述データの記述例を示す説明図である。

【図８】図１に示したディペンダビリティ維持システムおよび図４２に示した障害対応システムにおいて、ディペンダビリティ記述データとして用いる、D-Caseの基本構造を示す説明図である。

【図９】図５に示したディペンダビリティ記述データの一部のD-Case記述による表現を示す説明図である。

【図１０】図１に示したディペンダビリティ維持システムにおいて、ディペンダビリティ記述データが表現するシステムのディペンダビリティの価値を計測、評価する手法を示す説明図であり、（ａ）は各エッジを要素とする多次元ベクトル値をＤ値とする手法を示し、（ｂ）は有効エビデンス／総エビデンスをＤ値とする手法を示し、（ｃ）はモニタノード群のグラフ構造そのものをＤ値とする手法を示す。

【図１１】非特許文献５に記載の手法を示す説明図である。

【図１２】図１に示したディペンダビリティ維持システムのＤ値計算部の構成例を示す説明図である。

【図１３】図１に示したディペンダビリティ維持システムが処理する、ステークホルダ群により合意されたD-Case記述に電子署名を付与する例を示す説明図である。

【図１４】図１に示したディペンダビリティ維持システムが処理する、階層構造を有するモニタノードの例を示す説明図である。

【図１５】図１に示したディペンダビリティ維持システムが有する、変化対応サイクルおよび障害対応サイクルの概略を示す説明図である。

【図１６】図１５に示したディペンダビリティ維持システムが有する反復的プロセスを実現するためのアーキテクチャを示すブロック図である。

【図１７】図１に示したディペンダビリティ維持システムが有する、変化対応サイクルおよび障害対応サイクルの一連の手順を示すフローチャートである。

【図１８】図１７に示した変化対応サイクルにおける３つの処理を示すフローチャートで

ある。
【図１９】図１７に示した障害対応サイクルのうち、障害発生検出時の２つの処理を示すフローチャートである。
【図２０】図１９に示した障害発生検出処理の一例を示す図であり、（ａ）はディペンダビリティ記述データの一部を示し、（ｂ）はモニタノードを用いて障害検出するためのディペンダビリティ維持システムの構成例を示すブロック図である。
【図２１】図２０に示した障害発生検出処理において使用する、モニタノードと監視モジュールとの対応関係を定義するテーブルの一例を示す説明図である。
【図２２】図１９に示した障害発生検出処理において使用するスクリプトの構成概要を示す説明図である。
【図２３】図１７に示した障害対応サイクルのうち、障害予兆検出時の２つの処理を示すフローチャートである。
【図２４】図１９に示した障害発生検出処理において使用するスクリプトの一例を示す説明図である。
【図２５】図１に示したディペンダビリティ維持システムのランタイムコンピュータのソフトウェア階層の一例を示す説明図である。
【図２６】図１に示したディペンダビリティ維持システムのランタイムコンピュータの隔離部が実現する各隔離項目に関しての機能要件の一例をまとめたテーブルである。
【図２７】図１に示したディペンダビリティ維持システムにおけるディペンダビリティ記述データの変更抽出処理の手順を示すフローチャートである。
【図２８】図１に示したディペンダビリティ維持システムにおける、ディペンダビリティ記述データを介したワークスペースコンピュータとランタイムコンピュータとの連携を示す機能ブロック図である。
【図２９】図５に示したディペンダビリティ記述データを、計算機表現の一例としてＸＭＬで記述したリストの一部である。
【図３０】図１に示したディペンダビリティ維持システムの構成例であって、ディペンダビリティ記述データデータベースと、ワークスペースコンピュータおよびランタイムコンピュータとの関係を示す説明図である。
【図３１】図１に示したディペンダビリティ維持システムのランタイムコンピュータにおけるコマンド実行の処理手順を示すフローチャートである。
【図３２】図１に示したディペンダビリティ維持システムにおけるディペンダビリティ記述データに関連付けられたプログラムの処理内容の一例を示すフローチャートである。
【図３３】図１に示したディペンダビリティ維持システムにおけるディペンダビリティ記述データに関連付けられたプログラムの処理内容の他の例を示すフローチャートである。
【図３４】図１に示したディペンダビリティ維持システムのワークスペースコンピュータが備えるツール群の各機能と、ディペンダビリティ記述データとの関係を示す説明図である。
【図３５】図１に示したディペンダビリティ維持システムにおける、ベンチマーク機能とD-Case Editorとの連携を示す説明図である。
【図３６】図１に示したディペンダビリティ維持システムを２つ接続した構成例を示すブロック図である。
【図３７】図３６に示した２つのディペンダビリティ維持システムを接続した構成における、独立した２つのディペンダビリティ維持システム間での連携の一例を示す説明図である。
【図３８】図１に示したディペンダビリティ維持システムを２つ、本体側および部品側として統合した構成例を示すブロック図である。
【図３９】図３８に示した２つのディペンダビリティ維持システムを本体側および部品側として統合した構成における、独立した２つのディペンダビリティ維持システム間での連携の一例を示す説明図である。
【図４０】図３８に示した２つのディペンダビリティ維持システムを本体側および部品側

として統合した構成における、本体に部品を統合する処理手順を示すフローチャートである。
【図４１】図１に示したディペンダビリティ維持システムのワークスペースコンピュータによる、ディペンダビリティ記述データの表示例を示す説明図である。
【図４２】本発明の他の実施形態を示すものであり、障害対応システムの構成の概略を示す機能ブロック図である。
【図４３】図４２に示した障害対応システムにおいて、ディペンダビリティ記述データとして用いる、D-Caseの具体例を示す説明図である。
【図４４】図４２に示した障害対応システムの処理を示すフローチャートである。
【図４５】図４２に示した障害対応システムが用いる、D-Caseパターンを含むD-Caseの具体例を示す説明図である。
【図４６】図４２に示した障害対応システムが用いる、D-Caseパターン<=>モジュール対応テーブルの一例であって、（ａ）はモニタモジュールについての対応テーブルを示し、（ｂ）はアクションモジュールについての対応テーブルを示す。
【図４７】図４２に示した障害対応システムが用いる、D-Caseパターンを含むD-Caseの具体例を示す説明図である。
【図４８】図４２に示した障害対応システムが用いる、D-Caseパターン<=>モジュール対応テーブルの一例であって、（ａ）はモニタモジュールについての対応テーブルを示し、（ｂ）はアクションモジュールについての対応テーブルを示す。
【図４９】図４２に示した障害対応システムが用いる、D-Caseパターン<=>モジュール対応テーブルの一例であって、モニタモジュールについての対応テーブルを示す。
【図５０】図４２に示した障害対応システムにおいて、ディペンダビリティ記述データとして用いる、D-Caseの他の具体例を示す説明図（左半分）である。
【図５１】図４２に示した障害対応システムにおいて、ディペンダビリティ記述データとして用いる、D-Caseの他の具体例を示す説明図（右半分）である。
【図５２】図５０、図５１に示したD-CaseをＸＭＬ形式で記述した例を示す説明図であり、一つのサンプルからの抜粋を図５２～図５５に分割して示す。
【図５３】図５０、図５１に示したD-CaseをＸＭＬ形式で記述した例を示す説明図であり、一つのサンプルからの抜粋を図５２～図５５に分割して示す。
【図５４】図５０、図５１に示したD-CaseをＸＭＬ形式で記述した例を示す説明図であり、一つのサンプルからの抜粋を図５２～図５５に分割して示す。
【図５５】図５０、図５１に示したD-CaseをＸＭＬ形式で記述した例を示す説明図であり、一つのサンプルからの抜粋を図５２～図５５に分割して示す。
【図５６】図５０、図５１、および図５２～図５５に示したD-Caseから変換した監視制御データ（障害対応スクリプト）の例を示す説明図である。
【発明を実施するための形態】
【００４３】
〔実施の形態１〕
本実施の形態に係るワークスペースコンピュータ１０１および／またはランタイムコンピュータ１０２は、不完全性と不確実性が潜在的に存在するオープン環境において、対象システムのディペンダビリティの維持を支援するものである。そのため、後述するように、本実施の形態に係るワークスペースコンピュータ１０１および／またはランタイムコンピュータ１０２は、対象システムのディペンダビリティに関する仕様を記述したディペンダビリティ記述データ（時間軸上でモデルの変化を可能にした上で対象システムの構造を計算可能なモデルとして表現することを可能にする差分構造モデルで記述されるデータであってもよい。）に基づいて、対象システムのディペンダビリティの価値を定量的に示すＤ値を求める。
【００４４】
以下、図１から図４１に基づいて、本発明の一実施の形態について詳細に説明する。
【００４５】

〔１．ハードウェア構成〕
図１は、本実施の形態に係るディペンダビリティ維持システム１００の構成の概略を示す機能ブロック図である。
【００４６】
図１に示すように、ディペンダビリティ維持システム１００は、ディペンダビリティ維持装置としてのワークスペース（Workspace）コンピュータ（ディペンダビリティ維持装置、ワークスペース装置）１０１およびランタイム（Runtime）コンピュータ（ディペンダビリティ維持装置、ランタイム装置）１０２を含んで構成され、それらはネットワーク１０３で接続される。ワークスペースコンピュータ１０１で処理されたソフトウェアＳＷは、ランタイムコンピュータ１０２に展開されて処理される。また、対象システム、あるいはアプリケーションシステムとその実行に必要なライブラリ、ミドルウェア、システムサービス等のサポートシステムはランタイムコンピュータ１０２に含まれる。ソフトウェアＳＷは、バイナリ実行部１１０（１１０Ｗ、１１０Ｒ；ワークスペースコンピュータ１０１のバイナリ実行部１１０Ｗとランタイムコンピュータ１０２のバイナリ実行部１１０Ｒを区別する場合は、Ｗ，Ｒを付して記す。）で処理可能な表現形式を有している。ソフトウェアＳＷは対象システムで稼働するアプリケーションとその実行に必要なライブラリ、ミドルウェア、システムサービス等の実行環境を含む。
【００４７】
ワークスペースコンピュータ１０１は、検証ツール部１０１－０１、編集ツール部１０１－０２、解析ツール部１０１－０３、開発ツール部１０１－０４、合意形成支援ツール部１０１－０５、バイナリ実行部１１０Ｗを備えて構成される。なお、機能ブロックは必ずしもこれらに限定されるものではないが、これら代表的な機能について簡単に説明する。
【００４８】
検証ツール部１０１－０１は、ソフトウェアＳＷを検証する。編集ツール部１０１－０２は、ソフトウェアＳＷを編集する。解析ツール部１０１－０３は、ソフトウェアＳＷを解析する。その際、ランタイムコンピュータ１０２からの情報を用いても良い。開発ツール部１０１－０４は、ソフトウェアＳＷを開発するために利用される。
【００４９】
また、ランタイムコンピュータ１０２は、更新部１０２－０１、記録部１０２－０２、モニタリング部１０２－０３、再構成部（再構成手段）１０２－０４、隔離部１０２－０５、スクリプト処理部（スクリプト処理手段）１０２－０７、バイナリ実行部１１０Ｒを備えて構成される。なお、機能ブロックは必ずしもこれらに限定されるものではないが、これら代表的な機能について簡単に説明する。
【００５０】
更新部１０２－０１は、ランタイムコンピュータ１０２で処理されているソフトウェアＳＷを更新する。記録部１０２－０２は、ランタイムコンピュータ１０２の内部の状態を記録する。記録に当たっては、ソフトウェアＳＷの指示に従っても良いし、あらかじめ決められた所定の設定に従って記録しても良い。モニタリング部１０２－０３は、ランタイムコンピュータ１０２の内部の状態を取り出すと同時に、後述するＤ値を算出する。状態の取得は、ソフトウェアＳＷの指示に従っても良いし、あらかじめ決められた所定の設定に従って取得しても良い。再構成部１０２－０４は、ランタイムコンピュータ１０２の内部の構成を変更する。構成の変更に当たっては、ソフトウェアＳＷの指示に従っても良いし、あらかじめ決められた所定の設定に従って変更しても良いし、スクリプトの指示に従っても良い。隔離部１０２－０５は、ランタイムコンピュータ１０２の内部の一部の構成を切り離して独立させる。スクリプト処理部１０２－０７は、後述のディペンダビリティ記述データから導出されるスクリプトを実行する。
【００５１】
ここで、ディペンダビリティ維持システム１００は、図１に示すように、別個の二台のコンピュータで構成されるシステムであっても良いし、１台のコンピュータで構成される

システムであっても良いし、各々が２台あるいはそれ以上の台数のコンピュータから構成されるシステムであっても良い。２台あるいはそれ以上で構成されるときには、各コンピュータはネットワーク１０３で接続されていればよい。

【００５２】

図２に、ワークスペースコンピュータ１０１およびランタイムコンピュータ１０２のハードウェア構成を示す。もっとも基本的な構成としては、ワークスペースコンピュータ１０１およびランタイムコンピュータ１０２は、命令バスとデータバスで接続された演算装置１５１と、制御装置１５２と、メモリ装置１５３と、入出力装置１５４とを備えた電子計算機である。入出力装置１５４から入力されたビットデータの情報に基づき、演算装置１５１において、算術演算、論理演算、比較演算、シフト演算、等が実行される。実行されたデータは必要に応じて、メモリ装置１５３に記憶され、入出力装置１５４から出力される。これら一連の処理は、メモリ装置１５３に記憶されたソフトウェアプログラムに従って、制御装置１５２によって制御される。本実施の形態におけるワークスペースコンピュータ１０１およびランタイムコンピュータ１０２は、上記のコンピュータとしての基本機能を備えたハードウェアであり、オペレーティングシステムやデバイスドライバ、ミドルウェア、アプリケーションソフトウェアといったプログラム群によって制御されている。

【００５３】

図３に本実施の形態に係わるソフトウェアＳＷの構成例を示す。版管理情報１０４－０１は、当該ソフトウェアＳＷに関する版管理に関する情報であり、タイムスタンプを含んでいても良い。ディペンダビリティ記述データ１０４－０２は、当該ソフトウェアＳＷに関する後述のディペンダビリティ記述データに関する情報である。バイナリ・コード１０４－０３は、当該ソフトウェアＳＷの処理をバイナリ実行部１１０が解釈実行可能な表現形式で記述した情報である。外部参照情報１０４－０４は、当該ソフトウェアＳＷが必要とする、あるいは依存している外部のソフトウェアに関しての情報である。

【００５４】

〔２．ディペンダビリティ記述データ〕

図４は、一例としてのアプリケーションを示す説明図である。ここでは、WEBアプリケーションを例に説明する。

【００５５】

図４（ａ）は、当該アプリケーションが３層構造モデルを成していることを示している。プレゼンテーション層（Presentation）２００－０１は、当該アプリケーションの表示（及び入力）を担当する。データアクセス層（Data access）２００－０３は、当該アプリケーションが実行を進めるに当たって必要となるデータを格納し管理する。アプリケーションロジック層（Application logic）２００－０２は、ユーザからの入力の基づき当該アプリケーションとしての必要な計算処理を実行し、結果をデータアクセス層２００－０２に渡す。あるいは、当該アプリケーションが必要なデータをデータアクセス層２００－０２から入手し、処理をしてからプレゼンテーション層２００－０１に渡して利用者に提示する。

【００５６】

図４（ｂ）は、上記３層構造モデルを実装する際の一構成例を示している。WEB Server２０１－０２が、上記プレゼンテーション層２００－０１に相当する。Client２０１－０１は、当該アプリケーションの利用者であり、ネットワーク２０１－０５経由で、WEB Server２０１－０２を利用する。Client２０１－０１、ネットワーク２０１－０５、及びWEB Server２０１－０２を含めてプレゼンテーション層２００－０１に相当すると考えても良い。App Logic２０１－０３は、上記アプリケーションロジック層２００－０２に相当する。App Logic２０１－０３は、WEB Server２０１－０２とは通信路２０１－０６を介して情報の交換を行い、DBS２０１－０４とは通信路２０１－０７を介して情報の交換を行う。DBS２０１－０４は、データベースであり、App Logic２０１－０３の処理した、又は処理に必要なデータの格納、獲得、等の管理を行う。WEB Server２０１－０２、App Lo

gic２０１－０３、及びDBS２０１－０４は、ランタイムコンピュータ１０２で実行されるが、それらが１台のコンピュータ上で実行されても、それぞれ１台ずつのコンピュータ上で実行されても良い。

【００５７】

図５は、上記アプリケーションのディペンダビリティ記述データの一例であり、サービス継続の観点からディペンダビリティ記述データを記述している。ディペンダビリティ記述データとは、アプリケーションのディペンダビリティに関するステークホルダ間の合意された要求を記述したモデルである。

【００５８】

以下に、図５に記載の一例の内容を説明する。「WEBサービスの継続」（２０２－０１）は上記アプリケーションのディペンダビリティに関するゴールであり、WEBサービスが継続できることを意味している。このゴール（２０２－０１）を実現するためには、当該ディペンダビリティ記述データでは、「Clientの動作継続」（２０２－０２）、「WEB Serverの動作継続」（２０２－０３）、「App Logicの動作継続」（２０２－０４）、「DBSの動作継続」（２０２－０５）の４つの特性が満足される必要がある。それぞれ、図４（ｂ）のclient２０１－００１、WEB Server２０１－０２、App Logic２０１－０３、及びDBS２０１－０４に対応している。

【００５９】

これら４つの特性は、さらにブレークダウンされ次のように記述される。「Clientの動作継続」（２０２－０２）のためには、「デバイスの適切な動作」（２０２－０６）と「WEB Serverの適切な反応」（２０２－１０）という２つの特性の充足が必要である。「WEB Serverの動作継続」（２０２－０３）のためには、「リクエストの適切な送受信」（２０２－０７）と「App Logicの適切な反応」（２０２－１１）の２つの特性の充足が必要である。「App Logicの動作継続」（２０２－０４）のためには、「正常な業務処理」（２０２－０８）と「DBSの適切な反応」（２０２－１２）の２つの特性の充足が必要である。「DBSの動作継続」（２０２－０５）のためには、「DBの一貫性」（２０２－０９）と「データの有効性」（２０２－１３）の２つの特性の充足が必要である。

【００６０】

上記複数の特性の充足は、図５では、楕円形ノードに示された各種の検証を始めとする、実行時のモニタリング結果によって、把握、判断することができる。例えば、「デバイスの適切な動作」（２０２－０６）は包括的なテストによる「デバイス検査の合否」（２０２－２０）によって当該特性が充足されていると判断できる。「WEB Serverの適切な反応」（２０２－１０）や「リクエストの適切な送受信」（２０２－０７）は当該サーバへのアクセスベンチマークを始めとする、実行時のログ検証による「反応検査の合否」（２０２－２１）や「遅延検査の合否」（２０２－２２）によって当該特性が充足されていると判断できる。「App Logicの適切な反応」（２０２－１１）や「正常な業務処理」（２０２－０８）も同様に包括的なテストによる「テストケース合否」（２０２－２３と２０２－２４）によって当該特性が充足されていると判断できる。「DBSの適切な反応」（２０２－１２）、「DBの一貫性」（２０２－０９）、及び「データの有効性」（２０２－１３）の特性もベンチマーク、ストレステスト（過負荷試験）、アノマリーテスト（耐異常試験）による、あるいは実行時のデータベースアクセスログの検証による各々「ベンチマーク合否」（２０２－２５）、「ストレステスト合否」（２０２－２６）、「アノマリーテスト合否」（２０２－２７）によって当該特性が充足されていると判断できる。本実施の形態では、これら楕円形ノードに記述された検証を、対応ノードの特性が充足していることを示す根拠（エビデンス）として扱う。なお、これらのエビデンスは、後述するモニタノードによる計測値も利用される。

【００６１】

図６は、図５がシステムの正常系に関してのディペンダビリティ記述データであったのに対し、リスクを考慮したシナリオベースのディペンダビリティ記述データの例である。すなわち、「・・・が発生したら、・・・の対応が取れるか？」というシナリオを考える

ことで、ディペンダビリティ記述データを記述している。例えば、図６の例では、ノードのトップは「DBのHDDの容量が限界（データベースシステムを構成するハードディスク容量に空きがなくなったらどのような対応が取れるか？）」（２０３－０１）という、リスクが記述されている。このリスクが実際に発現しないための、４つの対策を次レベルのノードに記述している。「HDD容量の拡張」（２０３－０２）はハードディスクの追加やより容量の大きなディスクドライブへの置き換え等の手段によって容量が拡張可能であることを記述している。「バックアップDBSによる継続」（２０３－０３）はデータベースシステムにバックアップ系が用意されており、それを用いてデータベース機能が継続可能であることを記述している。「キャッシュによるApp Logicの継続」（２０３－０４）はデータベースが利用できない状況でも一部キャッシュされたデータを用いてアプリケーションが継続可能であることを記述している。「丁寧なエラーを利用者に返す」（２０３－０５）はアプリケーションの実行を続けられなくなっても何らかの説明を利用者に通知可能であることを記述している。

【００６２】

それらのノード（２０３－０２～２０３－０５）は、その可能性を裏付ける機能をそれらのサブノードで記述している。例えば、「HDD容量の拡張」（２０３－０２）は「HDDデバイス活線挿抜（ホットスワップ、すなわち停止させることなくディスクドライブを交換できること）」（２０３－０６）機能、あるいは「論理ボリュームのOSサポート（ボリューム容量はディスクドライブの容量に制限されないシステム機能）」（２０３－０７）機能によって実現可能になる。「バックアップDBSによる継続」（２０３－０３）は「スタンバイ機再起動（データベースシステムを複数用意しておき１つの系統が停止したときに待機していたシステムを起動させて機能を引き継ぐ）」（２０３－０８）機能、あるいは「２重系の片方のみでの継続（２系統のデータベースシステムで実現した機能をその１系統が停止しても継続できる）」（２０３－０９）機能によって実現可能になる。「キャッシュによるApp Logicの継続」（２０３－０４）は「DBへの変更の遅延書き込み（データベースへのアクセスをそれが復旧するまで遅らせる）」（２０３－１０）機能、あるいは「データのタイムスタンプ（タイムスタンプを参照することで古いデータでもApp Logicを継続できる）」（２０３－１１）機能によって実現可能になる。最後の「丁寧なエラーを利用者に返す」（２０３－０５）は「ネットワークが到達可能（利用者までのネットワーク通信が可能）」（２０３－１２）によって実現可能になる。

【００６３】

なお、これらエッジノード（２０３－０６～２０３－１２）は、図５における楕円形ノード（エビデンス）によってそれらの特性の充足が記述されるが、図６では省略している。

【００６４】

上記図５に記載のディペンダビリティ記述データと図６に記載のディペンダビリティ記載データは、どちらかをステークホルダ間で合意されたディペンダビリティ記述データとして用いても良い。また、より上位のスコープのディペンダビリティ記述データの一部として、図５は正常系、図６はリスク系として、共にステークホルダ間で合意されたディペンダビリティ記述データとして用いても良い。

【００６５】

図７は、ステークホルダ要求変更に伴って変化したディペンダビリティ記述データと、図６に示したディペンダビリティ記述データとの差分を示すディペンダビリティ記述データの記述例を示す説明図である。

【００６６】

図６では、「HDD容量の拡張」（２０３－０２）の可能性が当該アプリケーションの継続には重要であった。これに対して、図７では、何らかの環境の変化、例えば、ビジネスの好転によって、ステークホルダがデータベースシステムへの投資額を増やし強化する決定をした場合における、図６とは異なった時刻におけるディペンダビリティ記述データを記述している。すなわち、図６の「バックアップDBSによる継続」（２０３－０３）の代

わりに、「多重系によるDBSの無停止」（２０４－０１）機能を採用することを決定した場合を記述している。データベースが無停止であることによって、図６の「キャッシュによるApp Logicの継続」（２０３－０４）は不要になっている。その結果、図７の網かけされたノードが図６との差分となる。

【００６７】

図５から図７に示したディペンダビリティ記述データは、Safety Case表現（非特許文献２）を用いて記述しても良いし、図８、図９に示すように、D-Case表現を用いて記述しても良い。

【００６８】

ここで、図８を用いて、D-Case表現の基本構造を説明する。対象システムのディペンダビリティに関するステークホルダ群からの要求をトップゴール２１０－０１として記述する。トップゴールとは、対象システムに対してステークホルダ間で合意すべき命題を表す。例えば、「対象システムは機能安全規格IEC61508で定義されているSafety Integrity Level 3を満たす」などである。当該トップゴールが満たされていることを木構造により詳細化していき、詳細化されたサブゴール２１０－０５に対しても同様に木構造により詳細化していく。サブゴールとは、トップゴールを示すために、示すべき事を分割した命題を表す。サブゴールは、さらに小さなサブゴールに分割される。本実施の形態では、トップゴール及びサブゴールを総称してゴールと呼ぶ。当該詳細化の過程は「議論の構造」２１０－０２と呼ばれる。

【００６９】

当該ゴールをサブゴールに分割する際に、当該分割の補助説明（理由・理屈）を示すストラテジを記述しても良い。例えば、図５に記載のディペンダビリティ記述データの一部をD-Case記述で表現すると図９になる。ゴールが「ＷＥＢサービスの継続」（２０２－０１）であり、当該ゴールにリンクしている４項目である「Clientの動作継続」（２０２－０２）、「WEB Serverの動作継続」（２０２－０３）、「App Logicの動作継続」（２０２－０４）、「DBSの動作継続」（２０２－０５）は各々サブゴールであり、それら４項目をサブゴールとする理由がストラテジ「サブシステムの継続性の観点で分割」（２１１－０１）である。

【００７０】

また、ゴールあるいはストラテジにはコンテキストがリンクしても良い。コンテキストはゴールやストラテジの内容を補足する情報である。例えば、図９では、ゴール２０２－０１に対してコンテキストとして当該ゴール「WEBサービスの継続」において「継続」に関しての補足説明である「サービスが停止する場合でも１分以内で回復すること」が付与されている。

【００７１】

ゴールには、それが妥当であることを表明しているエビデンス２１０－０３、２１０－０４が存在する。エビデンスは分割され詳細化されたゴールを最終的に保証する。エビデンスの妥当性はステークホルダ間の合意に基づく。エビデンスの存在してないゴールは何らかの問題を内在している。

【００７２】

また、図５から図９では、ディペンダビリティ記述データを木構造で表現した。しかし、ノード間の依存関係を表現し一般的なグラフ構造で表現しても良いし、あるいは表形式で表現しても良い。以下、本実施の形態では、ディペンダビリティ記述データとして前記D-Case表現を用いて説明する。

【００７３】

本実施の形態では、ディペンダビリティ記述データから計算されるＤ値を用いてディペンダビリティの価値を計測、評価する。Ｄ値の計算は、ワークスペースコンピュータ１０１においては、検証ツール部１０１－０１のＤ値計算部（ディペンダビリティ値決定手段）１０１－０５が、ランタイムコンピュータ１０２においては、モニタリング部１０２－０３のＤ値計算部（ディペンダビリティ値決定手段）１０２－０６が行う。Ｄ値は、ワー

クスペースコンピュータ１０１においては、例えば、ディペンダビリティ記述データの検証のために用いられ、ディペンダビリティ記述データが変更される度に計算される。また、ランタイムコンピュータ１０２においては、例えば、ディペンダビリティ記述データに従って動作する対象システムの監視のために用いられ、対象システムの動作中はリアルタイムに計算される。

【００７４】

図１０にディペンダビリティ記述データが表現するシステムのディペンダビリティの価値を計測、評価する手法を示す。ディペンダビリティ維持システム１００にはディペンダビリティ記述データが複数存在する。トップノードは特定の特性に対するディペンダビリティの価値と定義できる。例えば、図５の「WEBサービスの継続」（２０２－０１）は失われると困るものであり資産（asset）と考えられるので、本実施の形態では「ディペンダビリティの価値」と定義し、「Ｄ値（評価値）」と呼ぶことにする。

【００７５】

トップノードは複数のサブノードに分解されるので、各エッジ（リンク）を要素とする多次元ベクトルが構成できる（図１０（ａ））。その際に、エッジに重みを付けても良い。図５を例にするなら、例えば、ノード２０２－０５をノード２０２－０４よりも重要である場合、２０２－０１と２０２－０５のエッジの重みを、２０２－０１と２０２－０４のエッジより大きくしても良い。また、あるノードのサブノードの数を重みとしても良い。サブノードが多いと言うことは十分に対策が検討されていることにもなるからである。このように重み付けされたエッジの多次元ベクトル値をＤ値とすることができる。次元数が多くなると多次元ベクトルは扱いにくいので、主成分分析法やフィッシャー判別分析法等を用いて次元削減を行っても良い。また、設計時におけるエビデンスを教師データとして学習し、後述のモニタノードによる変化を、例えばマハラノビス距離計算によって外れ値として検出することで異常の発生検知に用いても良い。

【００７６】

こうすることで、グラフ構造にエッジの重要性を加味した評価値が得られる。ここで得られるＤ値は多次元ベクトル値であるが、目的に応じた任意の変換手法によりスカラー値に変換できることはいうまでもない。

【００７７】

また、図１０（ｂ）は、当該特性において、有効エビデンス／総エビデンスをＤ値とした例である。図５では総エビデンス数は８であるが、例えば、その中で有効数が４であるときにはＤ値は４／８＝0.5となる。

【００７８】

また、図１０（ｃ）は、モニタノード群におけるモニタノードでの監視点が取得したデータが変動許容範囲内であるノードと当該変動許容範囲外のノードとを色分けし、当該グラフ構造そのものをＤ値とする例である。なお、詳細については後述する。

【００７９】

こうすることで、エビデンスの状態に応じた評価値が得られる。エビデンスは要求変更に応じて変化するため、ここで得られるＤ値も要求変更に応じて変化する。すなわち、Ｄ値を確認することで、要求変更がディペンダビリティに与える影響を評価することができる。

【００８０】

また、非特許文献５の手法を用いても良い。当該文献ではD-Case記述を図１１に記載の項目で再構成している。図１１に示すように、「フェーズ」、「目的」、「ターゲット」、「異常」の種類の４層にD-Caseノードを再配置することで、エッジの重みを各層で一貫性のある値として扱うことが可能になり、Ｄ値を各層で比較することも可能になる。

【００８１】

ここで、システムに求められるディペンダビリティ要件は、そのシステムの目的により異なる。また、システムのライフサイクルの各フェーズで、ステークホルダ間の合意のための作業を円滑化する必要があるが、そのためには、ディペンダビリティ要件やその実現

度合いを定量的に表現する必要がある。そこで、上記のように、Ｄ値としてディペンダビリティを定量的に表現することにより、議論するための指標（ディペンダビリティメトリクス）が実現できる。

【００８２】

上記のように、Ｄ値を用いることで、システム開発者は、そのシステムにおける特定のディペンダビリティ支援に関する重みを考慮して、要求そのもの、あるいは要求の実現度合いを定量化できる。また、システムの運用時には、そのシステムに不具合が発生した場合、その時点で満足し得るディペンダビリティをリアルタイムに定量化することができる。すなわち、ベンチマークや検証により得られるエビデンスとＤ値とを組み合わせて、ディペンダビリティ記述データを評価することが可能となる。また、動作中システムからリアルタイムにＤ値を得て、システムの状況を判断することが可能となる。

【００８３】

ここで、図１２を用いて、ランタイムコンピュータ１０２における、モニタリング部１０２－０３のＤ値計算部１０２－０６の一実施例を説明する。Ｄ値を当該ランタイムコンピュータ１０２においてステークホルダ合意に基づいて実行時に計算するために、図８に記載のD-Caseに対象システム内部のどの機能を、いつ、どのように監視するかをステークホルダ間での合意の上で記述するD-Caseモニタノードを導入する（図１２）。D-Case記述におけるゴールを表現するノードには、当該ゴールが満足しているエビデンスを表明するノードを関連付けることができる。この際のエビデンスに相当するノードに対象システム内部の監視点からのデータを収集し、当該データが変動許容範囲にあるかどうかを判断させるように構成できる。

【００８４】

変動許容範囲とは、監視点から得られたデータが基準値であるか、基準値に準じているかの範囲を示している。例えば、ネットワーク帯域の場合には、１Ｍｂｐｓ～２Ｍｂｐｓの範囲にあれば通常の動作であると見なすことができる範囲と定義できる。アプリケーションのメモリ消費であれば、１００ＭＢ以内の消費であれば通常の動作であると見なすことができる範囲と定義できる。なお、当該変動許容範囲はステークホルダ合意に基づいて設定されている。

【００８５】

図１２において、ゴール２２０－０１に対してはモニタノード２２０－０３が、ゴール２２０－０２に対してはモニタノード２２０－０４が、それぞれリンクしている。モニタノード２２０－０４は実行環境２２０－０７における監視点１（２２０－０５）からのデータを取得し、当該データが変動許容範囲にあるときにはゴール２２０－０２を満足していることを意味している。また、モニタノード２２０－０３は実行環境２２０－０７における監視点２（２２０－０６）からのデータを取得し、当該データが変動許容範囲にあるときにはゴール２２０－０３が満足していることを意味している。

【００８６】

ここで、監視点１（２２０－０５）の取得するデータはアプリケーション２２０－１０内部の監視対象センサ２２０－０８からデータを取得している。当該センサには、例えば、当該アプリケーションの消費しているメモリ量であったり、当該アプリケーションが外部と通信している遅延や帯域幅であったりする。一方、監視点２（２２０－０６）の取得するデータはアプリケーション２２０－１０内部の開始対象ログ２２０－０９から取得している。一般に、ログから対象システムの監視対象が変動許容範囲内にあることを判断するには、ログ内部を精査する必要がある。そこで、本実施例では一例として、スクリプトを導入し、ログを精査して監視点対象が変動許容範囲にあることを判断するように構成する。なお、当該スクリプトもステークホルダ合意に基づいている。

【００８７】

本発明の一実施例として、ステークホルダによる当該合意を電子署名によって表現し、実行時に確認するように構成しても良い。図１３にその一例を示す。図１２におけるD-Case記述の概略は図１３のように描けるが、ここでは３つの電子署名の有効範囲を示してい

る。範囲２１１－０１は全体に対する電子署名であり、範囲２２１－０２は１組のゴールとモニタノードに対する電子署名であり、範囲２２１－０３は１組のゴールとモニタノードとスクリプトに対する電子署名である。

【００８８】

前記電子署名は、その有効期限を定めるように構成可能である。例えば、開発時にのみ有効な電子署名をD-Case記述に付与したり、状況に依存した期間で有効な電子署名をD-Case記述に付与しても良い。これにより、モニタノードによるデータ取得の信頼性を高め、エビデンスとしての利用をより確実にできる。

【００８９】

また、電子署名のあるD-Case記述と電子署名のないD-Case記述とが混在する環境においては、電子署名のないD-Case記述を処理する場合にはデフォルトの動作を決めておくように構成できる。一般的にデフォルトの動作は、電子署名が無効である場合と同様の拒絶の処理をすべきである。しかし、状況によってはオペレータやユーザに確認を求めるように構成しても良い。

【００９０】

前記D-Caseモニタノードにて取得されるデータをＤ値計算に用いても良い。図１０（ｃ）に一例を示す。本実施例ではD-Case記述はモジュール毎に存在するように構成できる。当該D-Case記述そのものは木構造であり、その一部であるD-Caseモニタノードも木構造を構成するが、対象システム全体のD-Caseモニタノードの構造は図１０（ｃ）のような一般化されたグラフ構造を用いて構成できる。ここでは、ノード間の距離を評価するように構成しても良い。ノード群同士の直接リンクをノード間に複数のリンクがある構造よりも評価値を大きくできる。また、当該モニタノード群におけるモニタノードでの監視点が取得したデータが変動許容範囲内であるノードと当該変動許容範囲外のノードとを色分けし、当該グラフ構造そのものをＤ値として構成しても良い。当該色分けに際して、変動許容範囲に深刻度（Sevirity）を定義しても良い。例えば、ハードディスクの残容量に関する変動許容範囲の場合には、残容量５０％と残容量１０％とは異なった深刻度を与えた方が望ましい。当該グラフ構造にはモジュール間の依存関係を反映しているので、グラフ構造をＤ値とすることは、当該依存関係を考慮したＤ値をディペンダビリティの価値として扱うことができる。

【００９１】

さらに、図１４示すように、エッジのゴール群（２２０－０４乃至２２２－０７）以外に、途中のゴール群（２２２－０１乃至２２２－０３）に前記モニタノードをリンクしても良い（２２３－０１乃至２２３－０３）。あるゴールにリンクされたモニターノードと当該ゴールのサブノード群にリンクされたモニタノード群との関係は、対応する監視点群の関係に相当する。例えば、ゴール２（２２２－０２）が「商品購入サイトのサービス継続」である場合、そのサブゴールを２個に分割し、ゴール４（２２２－０４）が「電子決済システムの継続」であり、ゴール５（２２２－０５）が「商品ＤＢの継続」であるように構成できる。当該ゴール群に対して、各々次のモニタノードをリンクする。モニタノード２（２２３－０２）はゴール２（２２２－０２）にリンクされ、例えば、スクリプトによって包括的なサービス継続シナリオを実行することで、当該ゴール２が満足していることを確認する。モニタノード４（２２３－０４）はゴール４（２２２－０４）にリンクされ、例えば、スクリプトによって電子決済システムに異常が無いことを確認し当該ゴール４が満足していることを確認する。モニタノード５（２２３－０５）はゴール５（２２２－０５）にリンクされ、例えば、商品ＤＢに具備された死活センサからデータを受領することで当該ゴール５が満足していることを確認する。

【００９２】

また、次のように構成しても良い。例えば、ゴール３（２２２－０３）が「商品候補表示システムの継続」である場合、そのサブゴールを２個に分割し、ゴール５（２２２－０５）が「規定値アクセス数以下での正常動作」であり、ゴール６（２２２－０６）が「規定値遅延以内での正常動作」であるように構成しても良い。当該ゴール群に対して、各々

次のモニタノードをリンクする。モニタノード６（２２３－０６）はゴール６（２２２－０６）にリンクされ、例えば、スクリプトによって規定値アクセス数以下で正常に動作していることを確認することで、当該ゴール６が満足していることを確認する。モニタノード７（２２３－０７）はゴール７（２２２－０７）にリンクされ、例えば、対象システムのＷＥＢサーバに組み込まれたセンサのデータを取得して規定値以下の遅延で正常に動作していることを確認することでゴール７が満足していることを確認する。この場合には、ゴール３（２２２－０３）にリンクされたモニタノード３（２２３－０３）は、サブゴールである当該ゴール６と当該ゴール７が共に満足していることを確認することで、当該ゴール３が満足されていることを確認したこととするように構成できる。以上のように、モニタノード群をグラフ構造によって構成することで、上記Ｄ値をグラフ構造を計算することで取得し、システムの状況を判断することができる。

【００９３】

なお、本発明は以下のように構成してもよい。

【００９４】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、不完全性と不確実性が潜在的に存在するオープン環境におけるシステム開発及びシステム運用、さらにシステムが提供するサービスにおけるディペンダビリティを持続的に維持する目的で、かつ、ステークホルダ間における要求の誤解、環境の変化に対する対応不能、障害対応の失敗という３つの問題に対応する目的で、ディペンダビリティに関して時間軸上で変化可能なある時刻におけるディペンダビリティ記述データ（データ構造としては、例えば、木構造モデル）を備えるシステムにおいて、当該ディペンダビリティ記述データを計算することができ、それによって当該ディペンダビリティ記述データが表現するシステムのディペンダビリティの価値を計測、評価するものであってもよい。

【００９５】

このように、前記ディペンダビリティ維持装置は、ディペンダビリティに関して時間軸上で変化可能なある時刻におけるディペンダビリティ記述データを備えるシステムにおいて、ディペンダビリティの価値を計測、評価することができる。したがって、ステークホルダ間における要求の誤解、環境の変化に対する対応不能、障害対応の失敗という３つの問題に対処することでディペンダビリティを維持することが可能となるという効果を奏する。

【００９６】

また、本発明は以下のように構成してもよい。

【００９７】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、対象システムのディペンダビリティを維持するためのディペンダビリティ維持装置であって、対象システムのディペンダビリティに関する仕様を記述したディペンダビリティ記述データを取得する記述データ取得手段（ディペンダビリティ記述データ入力部９０１－０１、ソフトウェア入力部９０２－０１）と、上記記述データ取得手段が取得したディペンダビリティ記述データに基づいて、上記対象システムのディペンダビリティの価値を定量的に示す評価値（Ｄ値）を求めるディペンダビリティ値決定手段（Ｄ値計算部１０１－０５、Ｄ値計算部１０２－０６）と、を備えて構成されても良い。

【００９８】

これにより、対象システムのディペンダビリティの価値が定量的に表現できる。したがって、例えば、要求変更に伴ってディペンダビリティ記述データを変更するときや、対象システムの運用時に対象システムの状態を確認するときに、対象システムのディペンダビリティの価値を分かりやすく客観的に提示することができる。よって、対象システムのディペンダビリティの維持を円滑に行うことが可能となる。

【００９９】

さらに、本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、上記ディペンダビリティ記述データがノード間のエッジに重み付けが可能な木構造あるいはグラフ構造を有するデータであり、上記ディペンダビリティ値決定手段（Ｄ値計算部１０１－０５、Ｄ値計算部１０２－０６）は、上記エッジに重み付けされた重みを要素とする多次元ベクトル値を上記評価値（Ｄ値）とするように構成されても良い。

【０１００】

これにより、木構造あるいはグラフ構造に、エッジの重要性を加味した評価値が得られる。よって、評価値を確認することで、エッジに重み付けの変更がディペンダビリティに与える影響を評価することができる。なお、ここで得られる評価値は多次元ベクトル値であるが、目的に応じた任意の変換手法によりスカラー値に変換できることはいうまでもない。

【０１０１】

さらに、本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、上記ディペンダビリティ値決定手段（Ｄ値計算部１０１－０５、Ｄ値計算部１０２－０６）は、上記ディペンダビリティ記述データに含まれるエビデンスの総数である総エビデンス数に対する、上記対象システムをモニタして取得されたモニタ値が変動許容範囲に対して良好であったエビデンスの数である有効エビデンス数の割合を上記評価値（Ｄ値）として計算するように構成されても良い。

【０１０２】

これにより、エビデンスの状態に応じた評価値が得られる。エビデンスは要求変更に応じて変化するため、ここで得られる評価値も要求変更に応じて変化する。すなわち、評価値を確認することで、要求変更がディペンダビリティに与える影響を評価することができる。

【０１０３】

さらに、本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、上記ディペンダビリティ記述データをグラフ構造として評価し、上記ディペンダビリティ値決定手段（Ｄ値計算部１０１－０５、Ｄ値計算部１０２－０６）は、当該グラフ構造を計算して、例えばノード間距離や深刻度を計算式に組み込み評価値（Ｄ値）を算出しても良い。

【０１０４】

これにより、モニタノード群により得られるデータに応じた評価値が得られる。モニタノード群は対象システムの実行時の状況に応じて変化するため、ここで得られる評価値も当該実行状況に応じて変化する。すなわち、当該評価値を確認することで、実行状況がディペンダビリティに与える影響を評価することができる。

【０１０５】

また、本発明のディペンダビリティ維持システム１００は、対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムであって、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを仕様の変更に合わせて変更する変化対応サイクルＰ３００を実行するワークスペース装置（ワークスペースコンピュータ１０１）と、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルＰ３０１を実行するランタイム装置（ランタイムコンピュータ１０２）とを含み、上記ワークスペース装置および上記ランタイム装置の少なくともいずれか一方が、上記ディペンダビリティ記述データに基づいて、上記対象システムのディペンダビリティの価値を定量的に示す評価値（Ｄ値）を求めるように構成されても良い。

【０１０６】

さらに、本発明のディペンダビリティ維持システム１００は、他のディペンダビリティ維持システム１００とネットワークを介して接続されていても良い。すなわち、本発明の

ディペンダビリティ維持システム１００は、複数、ネットワークを介して接続されていても良い。

【０１０７】

〔３．二つのプロセス〕

図１５を用いて、ディペンダビリティ維持システム１００が有する反復的プロセスに関して、また、図１６を用いて、当該反復的プロセスを実現するためのアーキテクチャに関して説明する。

【０１０８】

オープンシステムディペンダビリティ、すなわち、機能、構造、システム境界が時間とともに変化し続けるシステムに対するディペンダビリティの実現の為には、反復的プロセスとしてのアプローチが必須である。当該反復的プロセスは、目的（Objectives）や環境の変化に対してシステムを継続的に変更していくためのサイクル（変化対応サイクルＰ３００）と、障害に対して迅速に対応するためのサイクル（障害対応サイクルＰ３０１）を備えている必要がある。当該反復的プロセスは、構成要素として要求マネジメントプロセス２５０－０１、開発プロセス２５０－０２、通常運用プロセスＳ３０２、障害対応サイクル２５０－０４、説明責任遂行プロセス２５０－０５等を含む「プロセスのプロセス（Process of Processes）」である。当該構成要素プロセスは相互に有機的に結びつけられる必要がある。本実施の形態では当該反復プロセスをDEOSプロセスと呼ぶ。

【０１０９】

対象システムのディペンダビリティに関する利害関係者をDEOSプロセスにおいては「ステークホルダ」と呼ぶ。ステークホルダとしては、以下を想定している。１）サービス・製品の利用者（顧客、社会的インフラの場合は社会全体）、２）サービス・製品の提供者（事業主）、３）システム提供者（設計開発者、保守運用者、ハードウェア供給者）、４）サービス・製品認可者（規制監督官庁）。

【０１１０】

ステークホルダは時間の経過や環境の変化によってそれぞれの目的を変化させ、機能やサービスに対する要求を変化させる可能性がある。これらの変化をここでは「目的・環境変化」と呼ぶこととする。これらの変化に対し、当該ステークホルダは熟慮し、相互に合意したうえで、適切な時期にシステムの変更を要求する。DEOSプロセスはこのような要求に対するサイクルとして、「変化対応サイクル」（Ｐ３００）を備える。

【０１１１】

対象システムは不完全さと不確実さに起因する障害を完全に回避することがきわめて困難である。障害の予兆を検出した場合には障害を未然に回避し、不幸にも障害が発生してしまった場合には迅速に対応して被害を最小化し、原因を究明し、説明責任を遂行する必要がある。DEOSプロセスではそのような状況に対応するために「障害対応サイクル」（Ｐ３０１）を備える。

【０１１２】

新規にシステムを開発し、また、目的・環境の変化に対応してシステムの変更を行う場合、その理由やステークホルダ間で行われた議論の過程、合意内容などを詳細に記録するための合意記述データベース２５０－０６を備えていることが効果的な前記反復的プロセスを実現し、説明責任を遂行するために必須である。当該データベースにはディペンダビリティを達成するための議論や根拠を記述した前記D-Case、障害に対してサービスを継続するためのシナリオを基に障害予兆の検出や障害発生に迅速に対応するための実行手続きを記述したスクリプト（詳しくは後述するが図中ではD-Scriptとして記載）が含まれている。そして、これらの合意記述を基に開発プロセス２５０－０２、障害対応サイクル２５０－０４、通常運用プロセスＳ３０２が実行され、また、説明責任遂行プロセス５０－０５を支援することができる。合意記述データベース２５０－０６は前記構成要素プロセスを有機的に繋ぐ重要な役割を果たす。

【０１１３】

前記DEOSプロセスの特徴をまとめる。１）「通常運用」（Ｓ３０２）から開始される「

変化対応サイクル（Ｐ３００）」と「障害対応サイクル（Ｐ３０１）」の２つのサイクルから成り立っていること。２）システム変更要求のための「ステークホルダ合意」（２５０－０７）と、システム変更や障害対応の「説明責任遂行」（２５０－０５）の２つのフェーズが組み入れられていること。３）ステークホルダ間で利害関係を調整し、ディペンダビリティを達成するために議論した過程、論拠、結果等を記述した前記「D-Case」と障害に迅速に対応するための実行手続きを記述した「スクリプト」を含む合意記述データベース２５０－０６を備え、構成要素プロセスを有機的に結合すること。

【０１１４】

図１５において通常運用Ｓ３０２はディペンダビリティ維持システム１００が通常運用状態にあることを示している。当該通常運用Ｓ３０２はシステムがステークホルダ間で合意されたサービスレベル変動許容範囲（In-Operation Range）から大幅な逸脱がなく、ユーザに対してサービス提供を継続している状態である。変化対応サイクルＰ３００は通常運用と並行して実行され、サービスの提供を継続しつつシステムの変更が行われることが望ましい。同様に、障害対応サイクルＰ３０１も通常運用を継続しながら実行されることが望ましい。実際、システムが異常の予兆を検知しても、当該スクリプトに記されたサービス・機能レベル変動許容範囲内で自動的に回避処理が働いてサービスが継続される場合がある。あるいは一部の機能を縮退してサービスを継続している場合もある。しかしながらサービスの提供が完全に停止されてしまう場合もある。

【０１１５】

通常運用状態Ｓ３０２において実行されるプロセスには、日常的な動作記録の点検、プロセスの定期的な見直し・改善、要員の訓練・しつけ・教育など、継続的なディペンダビリティ向上活動が含まれる。システムの稼働状況を記録し、日々点検する事により保守担当者や運用担当者が、何かの兆候をそこから見出す事ができる可能性がある。また、システムのメモリー資源を常にクリーンな状態にすることも、非常に有効な日常保守・改善活動である。あるいは、積極的に予行を行うことも有効である。障害はある時間が経過してある状態に達した時発生する。であれば、時間を先に経過させると障害の発生を事前に知ることができる。いわゆるリハーサルである。情報システムの提供するサービスの運用時において、どの程度適切なリハーサルができるのかはその時の状況による。

【０１１６】

障害対応サイクルＰ３０１に関して概説する。障害対応サイクルＰ３０１は障害に対して迅速に対応して障害による被害を最小化するためのサイクルである。DEOSプロセスでは「障害」をステークホルダ間で合意されたサービス・機能レベル変動許容範囲から逸脱する事象と定義する。障害対応サイクルにおける主要なフェーズは、「未然回避」（２５０－１０）、「迅速対応」（２５０－０９）、「原因究明」（２５０－０８）であり、障害が発生した場合は「説明責任遂行」が必須である。当該３フェーズはそれぞれ別個に、かつ順番に行われるとは限らない。多くの場合、これらはお互いが関連しあい、渾然一体となった事象・活動となる。

【０１１７】

未然回避フェーズ２５０－１０は、システムの運用中に障害が発生する前に障害発生を予知したり、あるいは障害が起きる可能性の増大を検出すると、障害を回避するように対応・動作するフェーズである。障害の予知が障害の発生予想時刻の充分に前であれば効果的な対策が打てる。例えばシステムの資源を制限してスループットを下げてシステムダウンを回避したりシステムダウンまでの時間を稼いだりすることが行われる。直前に予知した場合には障害の影響の最小化に努力することになる。また、原因解析に有効な、障害に至るまでのシステムの内部情報を記録することができる。予知のための具体的な方法としては、過去の障害パターンから類似の障害を判別することなどがある。未然回避シナリオはスクリプトとして事前に記述され、オペレータやシステム管理者と協調して未然回避動作が実行される。

【０１１８】

迅速対応フェーズ２５０－０９は、障害が起きた時にその影響を最小化するためのフェ

ーズである。障害に対する迅速対応のシナリオはスクリプトとして事前に記述されており、自動的に行われるのが望ましい。しかしながら、想定しない障害に対応しなければならない場面もある。対応分野や領域ごとの目的に応じたサービス継続のための緊急対応計画（責任者や対応組織、手順、エスカレーションパスなどが記されている）を事前に立てて、ステークホルダ間で合意しておくことが求められる。当該計画の指示に基づきオペレータやシステム管理者と協調して迅速に障害による影響を最小化することになる。すなわち、障害を分離して影響の局所化を行い、サービス全体のダウンを回避する。そのために障害が発生したアプリケーションやシステムの一部分のオペレーションを中断し、リセットし、その後にオペレータやシステム管理者による復帰活動が行われる。

【０１１９】

原因究明フェーズ２５０－０８は、障害対応サイクルＰ３０１と変化対応サイクルＰ３００に関連したフェーズである。サイクルにより深さの違う判断がなされる。障害対応サイクルＰ３０１では、どのような迅速対応が可能であるかを見極めることを目的とした原因究明がおこなわれる。その結果によっては変化対応サイクルＰ３００が開始される。

【０１２０】

説明責任遂行フェーズ２５０－０５では、サービス提供者、特に社会インフラサービス提供者や社会に広く使われる製品提供者が、障害発生時にサービス利用者、製品使用者、社会に対し、障害状況、迅速対応、今後の見通しなど説明する。これは利用者や社会からの信頼を維持し、インフラサービス提供上のコンセンサスを醸成し、ひいてはサービス提供者のビジネス遂行上の便益を守るという大変重要な役目を持つ。合意記述データベース特にD-Case記述と、システム監視記録が説明責任遂行に大いに役立つ。

【０１２１】

変化対応サイクルＰ３００に関して概説する。変化対応サイクルＰ３００はステークホルダの目的の変化や、各種外部環境の変化に対応するためのサイクルである。このサイクルにおける主要なフェーズは、システム変更のための「要求抽出・リスク分析」（２５０－１１）、「ステークホルダ合意」（２５０－０７）、「設計・実装・検証・テスト」（２５０－０２）である。大きな変化に対応する場合は「説明責任遂行」（２５０－０５）が必須となる。障害対応サイクルＰ３０１における原因究明フェーズ２５０－０８の実行の結果、システムの根本的な改良の要求が発生した場合も、変化対応サイクルＰ３００が開始される。

【０１２２】

要求抽出・リスク分析フェーズ２５０－１１は目的や環境の変化によりステークホルダからの要求が変化（新規の要求も含む）した場合、あるいは障害発生に迅速に対応した後、原因究明を行った結果、システムを変更する必要がある場合が始まる。いずれの場合も、事業主のサービス目的をベースにユーザの要求、システム環境、関連する法律や国際標準を勘案し、システムの機能要件を抽出する。また同時に、サービス目的からシステムのサービス継続シナリオを作成してリスク分析を行い、ディペンダビリティ要件を含む非機能要件を抽出する。

【０１２３】

ステークホルダ合意フェーズ２５０－０７では、何をどのように変更するのかを、すべてのステークホルダに分かりやすく、誤解のないように記述し、ステークホルダ間の議論を経て、合意をD-Caseとして記述する。またサービス継続シナリオを作成し、その実行手続きであるスクリプトを作成する。要求抽出・リスク分析フェーズ２５０－１１と当該ステークホルダ合意フェーズ２５０－０７が「要求マネジメントプロセス」（２５０－０１）を構成する。

【０１２４】

設計・実証・検証・テストフェーズ２５０－０２は、いわゆる設計開発のプロセスである。ここでは、これまで多くの研究がなされ、多くの手法やツールが出されている。

【０１２５】

説明責任遂行フェーズ２５０－０５では、目的や環境変化によるステークホルダの要求

変化を満たすためにシステムを変更した場合、その経緯と、いつからどのようにサービスや機能がよくなるのか（変化するのか）を説明する。また、日常のサービス遂行状況や設計開発・保守運用プロセスに関する説明が必要なときもこれに対応する。これは利用者や社会からの信頼を維持し、インフラサービス提供上のコンセンサスを醸成し、ひいてはサービス提供者のビジネス遂行上の便益を守るという大変重要な役目を持つ。合意記述データベース２５０－０６に記録されている特にD-Case記述が説明責任遂行に大いに役立つ。

【０１２６】

〔４．DEOSアーキテクチャ〕

前記DEOSプロセスは広範なオープンシステムに対するディペンダビリティを実現するための反復的プロセスを提供している。このプロセスをより具体的に対象とするシステムに適用した場合、対象のカテゴリー毎にプロセス実行のためのアーキテクチャを考える必要がある。本実施例では、組込みシステムを含む現代の大規模かつ複雑なソフトウェアシステムへの適応を念頭に考案されたアーキテクチャ（DEOSアーキテクチャと呼ぶ）について述べる。前記DEOSプロセスと当該DEOSアーキテクチャを並べて眺めるとDEOSプロセスが実際のシステムでどのように実行されるかが理解できる。

【０１２７】

図１６を用いて、当該DEOSアーキテクチャに関して説明する。当該アーキテクチャは次の構成要素から構成される。１）要求抽出・リスク分析フェーズを支援するツール群２６０－０１（１０１－０５）、２）ステークホルダ合意フェーズを支援する合意形成支援ツール群２６０－０２（１０１－０５）、３）合意の記述であるD-Caseとサービス継続シナリオの実行手続きであるスクリプト（図中ではD-Scriptとして記載）を含む合意記述データベース２５０－０６、４）DEOS実行環境２６０－０３、５）プログラム検証ツールとベンチマーキングならびにフォールトインジェクションテストのためのツール群を含むDEOS開発支援ツール２６０－０４。

【０１２８】

要求抽出・リスク分析フェーズ２５０－１１は事業主のサービス目的２６０－０５を基にユーザの要求、システム環境、関連する法律や国際標準を勘案し、システムの機能要件を抽出し、想定される障害に対するサービス継続シナリオを作成してリスク分析を行い、ディペンダビリティ要件を含む非機能要件を抽出する。

【０１２９】

ステークホルダ合意フェーズ２５０－０７は合意を形成するための方法と合意記述の記法に基づいて合意内容をD-Caseとして記述する。そのためのツールがD-Case Editor（２６０－０６）ならびにD-Case Viewer（２６０－０７）である。また、サービス継続シナリオに基づいた実行手続きスクリプト２６０－０８も作成される。当該スクリプトはDEOSアーキテクチャにおいてD-Case記述とアプリケーションプログラムの実行を動的に結合する役割を果たしている。当該スクリプトには後述のスクリプトエンジン（図中ではD-Script Engineとして記載）が実行するシナリオが書かれている。そのシナリオはDEOS実行環境２６０－０３に対して、１）いつ、どのようなログ情報を収集するかを指示し、また、２）障害発生時においては障害に対してどのように対処するかを指示している。この時、エスカレーションルールに従ってオペレータの介入を指示する場合もある。このように、当該スクリプトは動的かつ双方向に情報を交換することによりアプリケーションプログラムの実行を柔軟に制御し、オープンシステムディペンダビリティの達成に寄与している。

【０１３０】

DEOS実行環境２６０－０３はステークホルダ合意に基づくディペンダビリティを実現するサービスを提供するための実行環境であり、次のサブシステムから構成される。D-Visorは対象システムの再構成のため、システムの構成要素の各々の独立性を担保する仕組み（System Container）を提供する。あるSystem Container内における異常や障害が他のSystem Containerに波及することを抑える働きを担っている。D-Application Managerは複数のアプリケーションの独立性を担保する仕組み（Application Container）を提供し、各アプリケーションのライフサイクル（生成、起動、更新、停止、削除）を管理し制御す

る。D-Application Monitorはアプリケーションの動作監視機能を提供し、D-Caseモニタノードの記載に従ってエビデンスを収集し、D-Boxに蓄積する。D-System Monitorはシステム（オペレーティングシステムやランタイムサポートを含む）の動作監視機能を提供する。D-Application Monitor同様にD-Caseモニタノードの記載に従ってエビデンスを収集し、D-Boxに蓄積する。D-Boxはエビデンスを始め、OSD実現に有益な情報を安全・確実に記録する。D-Script EngineはD-Scriptを安全・確実に実行する役割を担い、D-Application Manager、D-Application Monitor、D-System Monitorを制御する。

【０１３１】

DEOS開発支援ツール２６０－０４は事業目的や事業継続シナリオに基づいて決められた機能仕様、テスト仕様、ベンチマーキングシナリオ、さらにはログ仕様に基づいてプログラムを設計し、開発し、検証し、ベンチマーキングを行い、テストを行うための開発支援ツール群である。例えば、型理論ならびにモデル検証に基づいたソフトウェア検証ツール、ベンチマーキング並びにフォルトインジェクション機能を備えたディペンダビリティテスト支援ツール等がある。

【０１３２】

前記DEOSプロセスを適用することにより享受できる最大の利点は、ステークホルダ間で要求の変化に対する合意議論を充分に行うことができ、合意結果や、その結論に至った理由や、議論の経緯をD-Caseに記録することができる点である。システム開発時にD-Case記述を用いることにより、前記DEOSアーキテクチャと連携して、障害時に適切かつ迅速な対応を取ることが可能なシステムを設計することができる。またD-Case記述があることにより、障害の原因究明や説明責任を果たすことがより容易になる。

【０１３３】

前記DEOSプロセスのもう一つの利点は、要求が適切に抽出されリスクが充分検討されたのちに、システムの変更が実行される点である。それぞれのステークホルダはシステムの状態をどんな時点でも、それぞれの観点で知ることができる。これによりシステムを簡潔かつ強力に管理運用することができる。一方、要求の数は膨大である。D-Case Editor（２６０－０６）やD-Case Viewer（２６０－０７）などのツールが要求マネジメントにおける作業を軽減する。

【０１３４】

前記DEOSアーキテクチャを具現化したDEOS実行環境２６０－０３は、モニタリング機能を備え、D-Caseを基に、解析に必要なシステムやアプリケーションの実行状態の監視と記録を実行する。 当該実行環境はこの監視記録とスクリプトに従って障害時の迅速対応を実行する。またD-Case記述や監視記録から得られた情報をエビデンスとして原因分析や説明責任を遂行する。スクリプトとスクリプトエンジンはD-Caseと当該実行環境の橋渡しを果たしており、この仕組みにより、システムの自動的、あるいは必要であればオペレータを介した（D-Case記述に基づく）柔軟な対応を可能にする。

【０１３５】

また前記DEOS-アーキテクチャは、あるモジュールにおける障害が他のモジュールに伝搬することを隔離する機能を提供する。同様にシステムのセキュリティを守るために外部からの侵入を検知する機能も提供する。またディペンダビリティ達成のための、実行前にプログラムを検証するツール、パフォーマンスを測定するツール、フォルトを埋め込んで異常時の振舞いをテストする開発ツール等も提供する。

【０１３６】

上記の仕組みや機能を利用することにより、前記DEOSプロセスと前記DEOSアーキテクチャは、継続的な障害回避のための能力を備え、障害時には適切かつ迅速な対応をして、その影響を最小限とする。またサービスを継続し、説明責任を遂行することができる。前記DEOSプロセスと前記DEOSアーキテクチャはオープンシステムディペンダビリティを達成するための初めての取り組みである。

【０１３７】

〔５．実行（ランタイム）環境〕

図１７を用いて本発明における一実施例に係る前記DEOSプロセスを構成する変化対応サイクルＰ３００と障害対応サイクルＰ３０１の一連の手順を説明する。当該両サイクル共に通常は通常運用Ｓ３０２の状態にある。

【０１３８】

変化対応サイクルＰ３００において、ビジネス環境の変化等の環境の変化によるステークホルダ要求の変更、開発途中における要求の変更、及び／又はシステム運用時における要求の変更が起こると、変化対応サイクルＰ３００は環境変化Ｓ３００－０１の状態に遷移する。

【０１３９】

まず、要求の変更Ｓ３００－０２をワークスペースコンピュータ１０１（図１参照）の解析ツール部１０１－０３を用いて行う。ここではステークホルダ群からの要求を抽出後に、当該要求の変更がランタイムコンピュータ１０２に与える影響を分析したり、具体的な変更手段を分析したり、変更箇所を分析したりする。

【０１４０】

次に、ディペンダビリティ記述データの変更Ｓ３００－０３を編集ツール部１０１－０２及び／又は開発ツール部１０１－０４を用いて、要求の変更Ｓ３００－０２での分析結果を基にして行う。ここでは、ソフトウェアＳＷにその結果が反映される。

【０１４１】

次に、変更が要求通りであることを、エビデンス計測Ｓ３００－０４にて、検証ツール部１０１－０１を用いて確認する。

【０１４２】

最後に、ソフトウェアＳＷをデプロイメントＳ３００－０５にてランタイムコンピュータ１０２に導入する。その後、変化対応サイクルＰ３００は通常運用Ｓ３０２になる。

【０１４３】

また、障害対応サイクルＰ３０１において、ランタイムコンピュータ１０２（図１参照）に障害が発生した場合には、障害発生Ｓ３０１－０１の状態に遷移する。まず、対応策抽出Ｓ３０１－０３においてランタイムコンピュータ１０２を停止させることなく、当該障害に対応する対応策を抽出する。

【０１４４】

次に、緊急対応Ｓ３０１－０４にて当該対応策を実行することで、ランタイムコンピュータ１０２におけるモデルに基づく処理を継続させる。

【０１４５】

次に、記録Ｓ３０１－０５では当該緊急対応を記録部１０２－０２により記録する。ここでは当該障害に関連する情報も記録部１０２－０２により記録される。ランタイムコンピュータ１０２はこの時点で継続されていることが望ましいが、障害によっては当該対応策でも継続が不可能な場合がある。継続・不継続のいずれの場合でも原因究明Ｓ３０１－０６を行う。これにより、当該障害の原因、および、ランタイムコンピュータ１０２がモデルに基づく処理が不継続の場合にはその原因が究明される。その結果次第では、要求の変更をともない、変化対応サイクルＰ３００を環境変化Ｓ３００－０１の状態に遷移させる。

【０１４６】

一方、障害対応サイクルＰ３０１において、故障（あるいは故障の予兆）が見つかった場合、あるいは障害の予兆が見つかった場合には、障害予兆Ｓ３０１－０２の状態に遷移する。故障（あるいは故障の予兆）の発見はランタイムコンピュータ１０２のモニタリング部１０２－０３によりなされる。障害の予兆の発見も当該モニタリング部１０２－０３によってなされる。具体的には、ランタイムコンピュータ１０２のバイナリ実行部１１０における各種の計算資源の消費履歴を調べることによって可能である。例えば、メモリ残量が一定時間内で減少を続けている場合には近い将来メモリ不足で計算が停止することがわかる。

【０１４７】

まず、対応策抽出Ｓ３０１－０３においてランタイムコンピュータ１０２における処理を停止させることなく、前記故障（あるいは故障の予兆）の発生、あるいは障害の予兆に対応する対応策を抽出する。

【０１４８】

次に、緊急対応Ｓ３０１－０４にて当該対応策を実行することで、ランタイムコンピュータ１０２を継続する。

【０１４９】

次に、記録Ｓ３０１－０５では当該緊急対応を記録部１０２－０２により記録する。ここでは当該障害に関連する情報も記録部１０２－０２により記録される。ランタイムコンピュータ１０２におけるモデルに基づく処理はこの時点で継続されていることが望ましいが、障害によっては当該対応策でも継続が不可能な場合がある。継続・不継続のいずれの場合でも原因究明Ｓ３０１－０６を行う。これにより、前記障害の原因、および、ランタイムコンピュータ１０２におけるモデルに基づく処理が不継続の場合にはその原因が究明される。その結果次第では、要求の変更をともない、変化対応サイクルＰ３００を環境変化Ｓ３００－０１の状態に遷移させる。

【０１５０】

なお、上述のように、前記変化対応サイクルＰ３００と障害対応サイクルＰ３０１は同時に存在する。

【０１５１】

なお、本発明は以下のように構成してもよい。

【０１５２】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００）は、前記ディペンダビリティ記述データを備えるシステムにおいて、環境の変化に起因するステークホルダの要求の変更、システム開発途中における要求の変更、及び／又はシステム運用時における要求の変更に対処するためのプロセスと、システム運用時におけるシステムが提供するサービスを障害発生に際しても継続するように対処し、及び／又はシステム運用時におけるシステムが提供するサービスを停止させないように障害の予兆を検知し障害の発生を未然に回避するプロセスという２つの手段を同時に備えていてもよい。

【０１５３】

このように、前記ディペンダビリティ維持装置は、不完全性と不確実性とを潜在的に有するオープン環境において、環境の変化に起因するステークホルダの要求の変更、システム開発途中における要求の変更、及び／又はシステム運用時における要求の変更に対処するためのプロセスと、システム運用時におけるシステムが提供するサービスを障害発生に際しても継続するように対処し、及び／又はシステム運用時におけるシステムが提供するサービスを停止させないように障害の予兆を検知し障害の発生を未然に回避するプロセスという２つの手段を備えることにより、従来の開発フェーズと運用フェーズの垣根を無くし、ステークホルダのアカウンタビリティ達成を可能にすると共に、ディペンダビリティを維持することが可能となる効果を奏する。

【０１５４】

また、本発明は以下のように構成してもよい。

【０１５５】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを仕様の変更に合わせて変更する変化対応サイクルＰ３００、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルＰ３０１の少なくとも何れか一方を実行するように構成されても良い。

【０１５６】

障害対応サイクルＰ３０１において、障害発生Ｓ３００－０１の状態に遷移することに

よる一連のステップは制御工学におけるフィードバック処理に相当し、障害予兆Ｓ３００－０２の状態に遷移することによる一連のステップは制御工学におけるフィードフォワード処理に相当する。従って、当該フィードバック処理の結果を学習することで、対応モデルを構築し、当該フィードフォワード処理に役立てるように構成することもできる。また、その際の当該モデルがディペンダビリティ記述データへの変更として記録することができるように構成しても良い。また、当該フィードフォワード処理においては、他のシステムにおける対応モデルを上記ディペンダビリティ記述データに記述することで、ステークホルダ群が合意した状態で当該モデルに対応する障害予兆に対してプロアクティブな回避処理を行うように構成することもできる。

【０１５７】

オープン環境において対象システムのディペンダビリティを継続的に維持するためには、要求変化や環境変化に対応して十分に検討して対処する変化対応サイクルＰ３００と、運用中の障害発生に迅速に対応する障害対応サイクルＰ３０１との両方が必要である。

【０１５８】

そこで、上記のように構成することにより、変化対応サイクルＰ３００および障害対応サイクルＰ３０１の少なくともいずれか一方において、対象システムのディペンダビリティの価値を示す評価値を求めることができる。したがって、例えば、変化対応サイクルＰ３００において、要求変更に伴ってディペンダビリティ記述データを変更するときに、評価値に基づいて、ディペンダビリティ記述データの変更案の適否を判断することが可能となる。また、例えば、障害対応サイクルＰ３０１において、対象システムの運用時に対象システムの状態を、評価値に基づいて判断することが可能となる。よって、対象システムのディペンダビリティの維持を円滑に行うことが可能となる。

【０１５９】

〔６．変化対応サイクル〕

図１８は、図１７に示したワークスペースコンピュータ１０１で作業される変化対応サイクルＰ３００における、ステークホルダ間での合意形成処理Ｓ４００、要求変更の実装処理Ｓ４０１、アカウンタビリティ処理Ｓ４０２の３つのステップを図示している。

【０１６０】

ステークホルダ間での合意形成処理Ｓ４００は要求の変更Ｓ３００－０２とディペンダビリティ記述データの変更Ｓ３００－０３と関係している。ステークホルダ間での合意形成処理Ｓ４００は、要求変更の内容を理解するステップ（要求変更の理解Ｓ４００－０１）から始まる。それには、例えば、“IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specification”に準拠したＳＲＳ（ソフトウェア内部仕様書）を分析する。

【０１６１】

次に、影響分析Ｓ４００－０２で既存のディペンダビリティ記述データに対する変更点を列挙する。

【０１６２】

最後に、当該変更点をディペンダビリティ記述データの変更Ｓ４００－０３において当該ディペンダビリティ記述データの変更として反映させる。

【０１６３】

これらのステップはステークホルダ間で合意Ｓ４００－０４されるまで繰り返される。

【０１６４】

また、要求変更の実装処理Ｓ４０１はディペンダビリティ記述データの変更Ｓ３００－０３とエビデンス計測Ｓ３００－０４と関係している。要求変更の実装処理Ｓ４０１は、設計Ｓ４０１－０１のステップから始まる。設計Ｓ４０１－０１では開発ツール部１０１－０４を利用して要求変更を実際のコードに変換するためのソフトウェア設計が、例えば能力成熟度モデル統合（Capability Maturity Model Integration, CMMI）に従って行われる。

【０１６５】

次に、実装Ｓ４０１－０２では、設計Ｓ４０１－０１での設計を開発ツール部１０１－０４を利用してソフトウェアＳＷに変換する。

【０１６６】

次に、テストＳ４０１－０３では、全ステップでのソフトウェアＳＷを検証ツール部１０１－０１を用いてテスト、検証する。

【０１６７】

これらのステップはテストＳ４０１－０３での検証が全てＯＫとなるまで繰り返される。検証が全てＯＫとなると、導入Ｓ４０１－０４で当該ソフトウェアＳＷをランタイムコンピュータ１０２に導入する。

【０１６８】

また、アカウンタビリティ処理Ｓ４０２はエビデンス計測Ｓ３００－０４とデプロイメントＳ３００－０５と関係している。アカウンタビリティＳ４０２は、ベンチマーク等収集Ｓ４０２－０１のステップから始まる。ベンチマークはランタイムコンピュータ１０２で実行され計測されても良いし、ランタイムコンピュータ１０２をシミュレーションする環境上で実行され計測されても良い。

【０１６９】

次に、収集されたベンチマークデータが要求を満足しているか検証される（エビデンス検証Ｓ４０２－０２）。

【０１７０】

最後に、ステークホルダは当該検証されたデータをエビデンスとして必要に応じて情報開示する（情報開示Ｓ４０２－０３）。なお、情報開示Ｓ４０２－０３により、ステークホルダがアカウンタビリティを達成（Ｓ４０２－０４）することができる。

【０１７１】

なお、本発明は以下のように構成してもよい。

【０１７２】

本発明のディペンダビリティ維持装置（ワークスペースコンピュータ１０１）は、上記環境の変化に起因するステークホルダの要求の変更、システム開発途中における要求の変更、及び／又はシステム運用時における要求の変更に対応するためのプロセスにおいて、それら要求を入力として、当該要求の変更の発生を起点とした一連のステップから構成されるループを形成する当該プロセスにおいて、当該要求の変更を確実に実装しシステム運用しサービスを提供できるようにステークホルダ間で要求の変更の合意を形成するステップと、当該要求の変更を実装し検証しシステム運用するステップと、当該要求の変更によるシステム運用とサービス提供によりステークホルダがアカウンタビリティを達成するステップという３つのステップを実行しても良い。

【０１７３】

〔７．障害対応サイクル（迅速対応）〕

図１９は、障害対応サイクルＰ３０１のうち、障害発生検出時の２つの処理を示すフローチャートである。なお、図１９における障害対応サイクルＰ３０１は図１７の再掲である。

【０１７４】

障害発生Ｓ３０１－０１はモニタリング部（障害発生検出手段）１０２－０３によって障害の発生が検出されると当該状態に遷移する。モニタリング部１０２－０３は、例えば、Ｄ値が所定の変動許容範囲より悪化したとき、障害発生を検出する。障害発生検出時の障害対応サイクルＰ３０１には２つの機能が関係している。システム再構成処理Ｓ５０２とアカウンタビリティ処理Ｓ５０３である。

【０１７５】

システム再構成処理Ｓ５０２は対応策抽出Ｓ３０１－０３と緊急対応Ｓ３０１－０４に関係している。システム再構成処理Ｓ５０２はサービス継続判断Ｓ５０２－０１から始まる。ここでは、障害に該当する部分のディペンダビリティ記述データを抽出する。当該ディペンダビリティ記述データからシステム再構成シナリオが抽出可能である場合には（Ｓ

５０２－０１でＹＥＳ）、次のステップに進む。なお、ディペンダビリティ記述データからシステム再構成シナリオが抽出不可能である場合には（Ｓ５０２－０１でＮＯ）、継続不可能と判断して、ディペンダビリティ記述データの変更抽出処理Ｓ８００（図２７）を実行する。

【０１７６】

システム再構成シナリオはシステム設計時点であらかじめ典型的なケースに関しては組み込んでおいても良い。例えば、後述のアプリケーション・コンテナあるいいはシステム・コンテナを再起動するシナリオはその一例である。また、当該再構成シナリオを後述同様にディペンダビリティ記述データにおけるノードに実行可能なプログラムを関連つけても良い。いずれの手法においても、当該再構成シナリオが抽出できない場合には、ディペンダビリティ記述データを変更するために、変化対応サイクルＰ３００を実行する準備をする。また、当該再構成シナリオを後述のスクリプトとして記述し、ディペンダビリティ記述データに関連つけても良い。

【０１７７】

次に、障害部位隔離Ｓ５０２－０２では当該抽出されたシナリオに従って障害部位を隔離し、ディペンダビリティ記述データの違いをランタイムコンピュータ１０２に反映させ、次ステップに進む。

【０１７８】

次に、サービス継続Ｓ５０２－０３では当該シナリオに従って隔離された部位の代替機能が当該シナリオによって起動され、サービスが継続される（Ｓ５０２－０４）。

【０１７９】

一方、アカウンタビリティ処理Ｓ５０３は記録Ｓ３０１－０５と原因究明Ｓ３０１－０６に関係している。アカウンタビリティ処理Ｓ５０３は障害に関係するログを記録する（Ｓ５０３－０１）ことから始まる。

【０１８０】

次に、前記シナリオに従って再構成されたランタイムコンピュータ１０２のエビデンスを収集する（Ｓ５０３－０２）。ここでのエビデンスとはシステム再構成によってランタイムコンピュータ１０２がディペンダビリティ記述データに従って適切に動作していることの記録である。

【０１８１】

次に、障害原因検出Ｓ５０３－０３に進む。ここでは、障害に関係する前記ログ（Ｓ５０３－０１）、前記エビデンス（Ｓ５０３－０２）、及び前記障害に関係したディペンダビリティ記述データをまとめたものである。本実施の形態では、当該まとめをステークホルダによるアカウンタビリティの達成とする。なお。これらのステップ（Ｓ５０３－０１及びＳ５０３－０２）は並列に処理されても良い。

【０１８２】

図２０にモニタリング部１０２－０３による障害発生検出処理の一例を示す。

【０１８３】

図２０（ａ）はディペンダビリティ記述データの一部（５１０）を示す。ディペンダビリティ記述データを作成する際に、実行時に状況が変化するデータは、モニタノードによって実行時にモニタリングが必要であることを明示しておく。

【０１８４】

ノード（５１０－０１）は「正常なＤＢＳのＨＤＤアクセス」に関する。当該ノードは２個に分岐され、すなわち、データベースシステムがＨＤＤ装置に対して正常にアクセスしている状態を２個のノードにブレークダウンして、その条件を記述している。具体的には、ノード（５１０－０２）は「ＨＤＤの残容量」に関する。また、ノード（５１０－０３）は「ＨＤＤの転送量」に関する。

【０１８５】

ここで、本例では、これら２ノードのエビデンスとして前記モニタノードを導入する。図２０（ａ）では、ノード（５１０－０４）とノード（５１０－０５）がモニタノードで

ある。ノード（５１０－０４）はＨＤＤの残容量をモニタする。ノード（５１０－０５）はＨＤＤ転送速度をモニタする。

【０１８６】

ディペンダビリティ記述データにおけるモニタノードは、対象のシステムを外部からモニタリング可能な場合にはモニタリング部１０２－０３が備える監視モジュール群（後述）を利用して対象のシステムをモニタリングする。一方、対象のシステムを外部からモニタリング不可能な場合には、対象のシステムに後述のプログラムを導入する手順に従って監視モジュールを送り込むことでモニタリングを行う。

【０１８７】

そして、実行時のモニタリング結果をディペンダビリティ記述データの記述と比較することにより、変動許容範囲を超える差分の場合には、設計時の基準と異なる振る舞いとして異常として特定する。また、ディペンダビリティ記述データの上位ノードを参照することで、複合的な異常により上位ノードのディペンダビリティが満たされない状況を把握できる。さらに、個々のシステム内や複数のシステム間の障害箇所を突き合わせることにより、障害発生箇所を絞り込むことができる。

【０１８８】

図２０（ｂ）にモニタノードを用いて障害検出するためのディペンダビリティ維持システム１００の構成例を示す。モニタノードはパターンによって分類することができる。例えば、図２０（ａ）の２個のモニタノード（５１０－０４、５１０－０５）は『HDD「＄１」の「＄２」が「＄３」であることをモニタリングする』という形式を取ることができる。このモニタノードのパターンに対応するように監視モジュール群を予め用意し、モニタノードのパターンと監視モジュールの対応関係をテーブル化しておくことにより、実行時の状態をモニタリングできる。

【０１８９】

例えば、図２０（ａ）の２個のモニタノード（５１０－０４、５１０－０５）に関しては、図２１のように対応関係を示すことができる。ここで、図２１は、『HDD「＄１」の「＄２」が「＄３」であることをモニタリングする』というパターンのモニタノードの場合に、＄２が「ディスク容量」であれば、監視モジュールとしてHDD監視モジュールを利用し、引数として＄１（＝HDD識別番号）と＄３（＝変動許容範囲）を指定すれば良いことを示している。

【０１９０】

ワークスペースコンピュータ１０１の編集ツール部１０１－０２はディペンダビリティ記述データＭ１を記述し、さらに、ランタイムコンピュータ１０２における内部表現であるディペンダビリティ記述データ内部表現Ｍ２に変換する。詳細には、ワークスペースコンピュータ１０１の編集ツール部１０１－０２において、ディペンダビリティ記述データ記述・表示ツール５１１－０１がディペンダビリティ記述データＭ１を記述した後、それを、ディペンダビリティ記述データ変換部５１１－０３によってディペンダビリティ記述データ内部表現Ｍ２に変換する。

【０１９１】

ランタイムコンピュータ１０２では、ワークスペースコンピュータ１０１から取得したディペンダビリティ記述データ内部表現Ｍ２を更新部１０２－０１がディペンダビリティ記述データ記録部（ディペンダビリティ記述データ格納部）５１１－０５に記録する。また、更新部１０２－０１は、ディペンダビリティ記述データ内部表現Ｍ２に基づいて、モニタノードとモニタリング部１０２－０３における各種監視モジュール（監視モジュール群５１１－ｍ）との対応関係をモニタノード／モジュール対応テーブル５１１－０６に記録する。なお、図２０（ｂ）には監視モジュール群５１１－ｍの各監視モジュールとして、ＣＰＵ監視モジュール５１１－０８、ＨＤＤ監視モジュール５１１－０９、プロセス監視モジュール５１１－１０、が記載されているが、他の監視モジュール５１１－１１があっても良い。

【０１９２】

モニタリング部１０２－０３における前記監視モジュールによって得られたデータ（モニタ値）は収集部５１０－１３によって集められ、実行状況検証部５１０－１２によって、モニタノード／モジュール対応テーブル５１１－０６に基づき、前記ディペンダビリティ記述データにおけるモニタノードの値として記録される。例えば、ディペンダビリティ記述データの各モニタノードに前記モニタ値の変動許容範囲を対応付けて記載しておき、モニタ値がその変動許容範囲から外れたノードを障害発生ノードとして検出する。また、前記監視モジュール群５１１－ｍからのモニタ値はＤ値計算部１０２－０６においてノードに対するＤ値の計算に利用される。よって、例えば図２０（ａ）のディペンダビリティ記述データの一部５１０では、２個のモニタノード（５１０－０４、５１０－０５）から取得されたモニタ値が変動許容範囲と比較されるとともに、これに基づいてＤ値が算出される。

【０１９３】

また、ＣＰＵ利用率、メモリ使用量、プロセス死活等は、のＯＳ機能としてあらかじめ備わっており、外部からモニタリング可能なものもあるが、「システムＡ内のある処理Ｘが正常終了した」というような、外部からのモニタリングが難しいものもある。これらは、主に、アプリケーション固有の情報であり、ランタイムコンピュータ１０２に監視モジュール５１１－１４として組み込む。すなわち、モニタリング対象をモニタリングする監視モジュールが内部にない場合、監視モジュール５１１－１４のように、外部に監視モジュールを設けることができる。この組み込みはシステム構築時であっても良いし、ランタイムコンピュータ１０２の実行時であっても良い。これにより、実行時のモニタリング結果が変動許容範囲を超えた場合に、ディペンダビリティ記述データのノード単位で異常を特定できる。

【０１９４】

また、実行時に各々の監視モジュールからモニタリング結果を集める（５１０－０４、５１０－０５）ことにより、ディペンダビリティ記述データ上のどのモニタノードで変動許容範囲を超えているかを判断できる。つまり、ディペンダビリティの観点から問題の発生を把握できる。すなわち、ディペンダビリティ記述データを当該モニタノードから上位側へ検討することで、どの範囲でディペンダビリティが維持できなくなったかを判断できる。例えば、ノード（５１０－０４）及びノード（５１０－０５）のモニタ値が共に変動許容範囲を超えているような状況では、ディペンダビリティを維持できなくなった原因がＨＤＤ以外にあるとして、ディペンダビリティ記述データのより上位を探索する。さらに、複数のシステムが相互作用している状況では各々のシステムに関するディペンダビリティ記述データだけではなく、相互作用に関するディペンダビリティ記述データも記述されていれば、適切なモニタノードによってシステム相互間で変動許容範囲を超える振る舞いの部位を絞り込むこと、あるいは特定することができる。

【０１９５】

前記監視モジュール群５１１－ｍはシステムにあらかじめ組み込まれた監視モジュール群であるが、運用中に新たな監視モジュールを組み込む場合、あるいは変動許容範囲の判断が複数のセンサからのデータを考慮しなければ行けなかったり、ログからのデータを解析しなければ行けなかったりする場合には、スクリプトで構成することができる。

【０１９６】

図２２を用いて、スクリプトの構成概要を説明する。スクリプトは１以上の基本動作５５０－０１（図ではD-Taskとして記載）と１以上の制御動作５５０－０２（図ではD-Controlとして記載）から構成される。制御動作５５０－０２は変動許容範囲を深刻度で分けて該当する深刻度の基本動作（D-Task）を実行する。例えば、ハードディスクの残容量の場合には、５０％の残容量の深刻度を「低」、７５％の残容量の深刻度を「中」、９０％の残容量の深刻度を「高」として設定して、制御動作５５０－０２（D-Control）を構成できる。この場合、深刻度「高」の場合にはD-Task 1（５５０－０３）を実行し、深刻度「中」の場合にはD-Task 2（５５０－０４）を実行し、深刻度「低」の場合にはD-Task 3（５５０－０５）を実行する。

【０１９７】
なお、本発明は以下のように構成してもよい。
【０１９８】
本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスを障害発生に際しても継続するように対処するプロセスにおいて、障害の発生を検出する手段（モニタリング機能）と、障害の発生した部分を隔離する手段（隔離機能）と、隔離された残りの部分でサービスを継続する手段と、障害の原因を検出する手段（原因追及機能）と、上記システム運用時におけるシステムが提供するサービスの継続性を判断する手段（継続性判断機能）とを備えるプロセスにおいて、当該障害の発生を起点とした一連のステップから構成されるループを形成する当該プロセスにおいて、当該障害の発生に際してサービスの停止期間を最小限にするように前記ディペンダビリティ記述データを備える対象システムを再構成するステップと、当該再構成によるシステム運用とサービス提供によりステークホルダがアカウンタビリティを達成することができるステップという２つの処理を実行しても良い。
【０１９９】
また、本発明は以下のように構成してもよい。
【０２００】
本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを仕様の変更に合わせて変更する変化対応サイクルＰ３００、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルＰ３０１のうち、少なくとも上記障害対応サイクルＰ３０１を実行するとともに、上記障害発生を検知したとき、上記ディペンダビリティ記述データに基づいて、当該対象システムを再構成する再構成手段（再構成部１０２－０４）を備えて構成されても良い。
【０２０１】
これにより、障害対応サイクルＰ３０１を実行するランタイムコンピュータ１０２において、対象システムの障害発生を検知したとき、ディペンダビリティ記述データに基づいて、当該対象システムを再構成して、対象システムの運用を継続させることができる。
【０２０２】
そして、ランタイムコンピュータ１０２は、対象システムの障害発生を、Ｄ値に基づいて検知してもよい。例えば、Ｄ値が所定の変動許容範囲より悪化したとき、障害発生を検出するようにしても良い。
【０２０３】
〔８．障害対応サイクル（障害未然回避）〕
図２３は、障害対応サイクルＰ３０１のうち、障害予兆検出時の２つの処理を示すフローチャートである。なお、図２３における障害対応サイクルＰ３０１は図１７の再掲である。
【０２０４】
障害予兆Ｓ３０１－０２はモニタリング部（障害予兆検出手段）１０２－０３によって障害の予兆、あるいは故障（あるいは故障の予兆）が検出されると当該状態に遷移する。なお、障害の予兆に関しては各種の手法を組み合わせて実現可能である。例えば、ハードディスクドライブ装置はS.M.A.R.T.の仕組みを使うことで障害の予兆を検出できる。また、モニタリング部１０２－０３は、例えば、Ｄ値の時間変化の傾向が所定の基準より悪化したとき、障害予兆を検出する。また、当該モニタリング部１０２－０３は、前記モニタノードでモニタリングされているデータの時間変化の傾向が変動許容範囲を超えている場合に障害予兆として検出する。障害予兆検出時の障害対応サイクルＰ３０１には２つの機能が関係している。システム再構成処理Ｓ６０２とアカウンタビリティ処理Ｓ６０３である。

【０２０５】
システム再構成処理Ｓ６０２は対応策抽出Ｓ３０１－０３と緊急対応Ｓ３０１－０４に関係している。システム再構成処理Ｓ６０２は障害部位とその影響範囲の同定Ｓ６０２－０１から始まる。ここでは、障害の予兆と障害そのものの検出を説明の都合上同一視して障害として扱う。まず、ディペンダビリティ記述データのどの部位が障害かを同定する。そして当該ディペンダビリティ記述データからその影響範囲を同定する。影響範囲はディペンダビリティ記述データのノードを探索することにより同定する。例えば、障害部位に対応するノードが複数ある場合には、それらの共通ノードまでを影響範囲として特定できる。また、ノードにノード間の依存関係が記録されている場合には、当該依存関係を利用しても良い。

【０２０６】
次に、サービス継続判断Ｓ６０２－０２では、当該ディペンダビリティ記述データから抽出される対応シナリオが当該影響範囲を十分にカバーしてサービスが継続できるか、判断される。継続可能の場合には（Ｓ６０２－０２でＹＥＳ）、障害部位隔離Ｓ６０２－０３に進む。なお、ディペンダビリティ記述データから対応シナリオが抽出できない場合には（Ｓ６０２－０２でＮＯ）、継続不可能と判断して、ディペンダビリティ記述データの変更抽出処理Ｓ８００（図２７）を実行する。当該変更抽出処理Ｓ８００は、サービス継続判断Ｓ６０２－０２にて継続可能の場合であっても、当該障害予兆の内容によっては実行された方が良い状況もあるので、一実施例として当該変更抽出処理を実行するように構成しても良い。

【０２０７】
次に、障害部位隔離Ｓ６０２－０３では、抽出された対応シナリオに従って障害部位を隔離する。

【０２０８】
次に、サービス継続Ｓ６０２－０４に進み、システム再構成処理Ｓ６０２を終了する。この時点でランタイムコンピュータ１０２によるサービスは継続されている（Ｓ６０２－０５）。

【０２０９】
一方、アカウンタビリティ処理Ｓ６０３は記録Ｓ３０１－０５と原因究明Ｓ３０１－０６に関係している。アカウンタビリティ処理Ｓ６０３は内部状態の記録Ｓ６０３から始まる。ここでは、前記障害に関連したディペンダビリティ記述データ、及び当該ディペンダビリティ記述データに関連したランタイムコンピュータ１０２の内部の状態を記録する。

【０２１０】
次に、システム再構成処理Ｓ６０２後のエビデンスを収集する（Ｓ６０３－０２）。ここでのエビデンスとはシステム再構成によってランタイムコンピュータ１０２がディペンダビリティ記述データに従って適切に動作していることの記録である。なお、本実施の形態では、当該エビデンスの収集をもってステークホルダによるアカウンタビリティの達成とする（Ｓ６０３－０３）。

【０２１１】
前記緊急対応Ｓ３０１－０４は上記スクリプト（D-Script）を用いて処理しても良い。図２４に一例を示す。当該スクリプト（D-Script）は図４（ｂ）記載のWEBサービスの再起動のシナリオ例であり、６個の基本動作（D-Task）から構成されている。当該シナリオは上記DEOSプロセスにおける要求抽出・リスク分析フェーズ２５０－１１におけるサービス継続シナリオから導出される。ステークホルダは当該シナリオに関して合意している。当該サービス継続シナリオは要求抽出・リスク分析フェーズを支援するツール群２６０－０１を用いて記述され、図２４の形式で記述され、スクリプトに変換され、ステークホルダ合意によって電子署名される。

【０２１２】
スクリプトがラインタイムコンピュータ１０２内部の後述のスクリプトエンジン部７１０（１０２－０７）において実行される際には、当該スクリプトに附随された電子署名が

検証される。電子署名が有効でない場合にはスクリプトは実行されない。さらに、スクリプトの実行に際しては、その権限を確認するように構成しても良い。すなわち、誰がどのスクリプトの操作を実行可能かを確認するように構成可能である。その場合には権限を有しない状況では当該スクリプトは実行されない。さらに、スクリプト自体を暗号化してランタイムコンピュータ１０２内部に格納するように構成しても良い。これらのセキュリティに関する施策により、当該スクリプトが原因となるセキュリティ違反を防止することができる。

【０２１３】

図２５を用いてランタイムコンピュータ１０２に関するソフトウェアの階層構造の一例を説明する。

【０２１４】

図２５では３個のＣＰＵコア７０１（７０１－０１～７０１－０３）を有するマルチコア構成例を示している。なお、図２５では周辺デバイスは省略している。ＣＰＵコアの上位層にはシステム・コンテナ提供部７０２がある。これは、隔離部１０２－０５（図１）の一部であり、上位層に対して複数のＯＳカーネル７１１を動作可能にする、隔離されたコンテナを提供する。また、図２５では各々のＣＰＵコア７０１に対してそれぞれ１つのシステム・コンテナ７１６、７１８、及び７２０が割り当てられているが、複数のシステム・コンテナを１つのＣＰＵコア７０１に割り当てても良いし、１つのシステム・コンテナに対して複数のＣＰＵコア７０１…を割り当てても良い。

【０２１５】

ＣＰＵコア７０１－０１にはシステム・コンテナ７１６を割り当て、そこではＯＳカーネル７１１上に、システムモニタ部７０４、再構成部１０２－０４、システム記録部７０６、ディペンダビリティ記述データ処理部７０７が配置される。

【０２１６】

システム・コンテナ７１６はランタイムコンピュータ１０２に１つ存在する。システムモニタ部７０４はモニタリング部１０２－０３（図１）の一部でありランタイムコンピュータ１０２におけるシステム機能をモニタする。特に他のシステム・コンテナ（図２５ではシステム・コンテナ７１８と７２０）のＯＳカーネル７１１をモニタする。また、当該システムモニタ部７０４に学習機能を持たせるように構成しても良い。例えば、対象システムにおけるシステム機能をモニタする際に、そのデータ構造に対する整合性をモニタするように構成する場合に、当該システム機能に対応した仕様書やソースコードから整合性のための条件を自動生成するように構成することができる。実行時に当該条件を学習することで、当該データ構造の全てをモニタしないでも、または、当該データ構造が変更されても当該モニタの仕組みを変えないでモニタ機能を継続することができる。

【０２１７】

再構成部１０２－０４は、他のシステム・コンテナの再構成を行う。具体的には、既に存在するシステム・コンテナを停止させ、破壊し、新しい構成でシステム・コンテナを作り出し、当該構成に従ってＯＳカーネルを始めとする機能を配置し起動する。システム・コンテナが破壊されるときには、当該システム・コンテナの内部状態を、システム記録部７０６を用いて記録しても良い。システム記録部７０６は記録部１０２－０２（図１）の一部であり、ランタイムコンピュータ１０２の内部状態を記録する。

【０２１８】

ディペンダビリティ記述データ処理部７０７は更新部１０２－０１（図１）の一部であり、ワークスペースコンピュータ１０１で生成されたソフトウェアＳＷの内部にあるディペンダビリティ記述データをランタイムコンピュータ１０２全体のディペンダビリティ記述データに取り込み、ディペンダビリティ記述データを処理するためのシナリオを抽出しランタイムコンピュータ１０２を、再構成部１０２－０４を利用して再構成する。

【０２１９】

一方、ＣＰＵコア７０１－０２にはシステム・コンテナ７１８を割り当て、そこではＯＳカーネル７１１上に、アプリケーションモニタ部７０８、アプリケーション記録部７０

９、スクリプトエンジン部７１０、アプリケーション・コンテナ提供部７１２、アプリケーション管理部７１３乃至７１４、そしてアプリケーション群を配置している。

【０２２０】

ここで、アプリケーション・コンテナとは、前記システム・コンテナ１個に対して１個以上存在し、１個以上のアプリケーションをグループ化し、アドレス空間の独立、名前空間の独立、及びＣＰＵスケジューリングの独立を実現し、アプリケーション・コンテナ提供部７１２により提供される。図２５ではシステム・コンテナ７１８にアプリケーション・コンテナ７１７及び７１９の２個を配置している。

【０２２１】

アプリケーションモニタ部７０８はモニタリング部１０２－０３（図１）の一部であり、アプリケーション・コンテナ内部をモニタする。アプリケーション記録部７０９は記録部１０２－０２（図１）の一部であり、アプリケーション・コンテナの内部状態を記録している。

【０２２２】

アプリケーション・コンテナ７１７にはアプリケーション管理部７１３を配置し、１個以上のアプリケーションが利用する。同様にアプリケーション・コンテナ７１９内部にはアプリケーション管理部７１４を配置し、１個以上のアプリケーションが利用する。

【０２２３】

システム・コンテナ７２０の内部構成はシステム・コンテナ７１８と同様である。そのため、説明および図２５中の表示を省略する。

【０２２４】

図２６にランタイムコンピュータ１０２の隔離部１０２－０５（図１）が実現する各隔離項目に関しての機能要件の一例をまとめる。これらの隔離項目は一部を選択して必要な機能要件のみを実現しても良い。例えば、前記システム・コンテナは、これら項目の全部が実現されているし、アプリケーション・コンテナはアドレス空間、名前空間、及びＣＰＵスケジューリングの３項目が実現されている。

【０２２５】

なお、本発明は以下のように構成してもよい。

【０２２６】

本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスを停止させないように障害の予兆を検知し障害の発生を未然に回避するプロセスにおいて、前記ディペンダビリティ記述データを備えるシステムの内部状態を記録する手段（ロギング）と、当該記録された内部状態から前記ディペンダビリティ記述データを備えるシステムの障害の予兆を検出する手段と、前記ディペンダビリティ記述データを備えるシステムの当該検出された障害に対応する部分を同定する手段と、当該検出された障害が上記システム運用時におけるシステムが提供するサービスの継続性を判断する手段とを備えるプロセスにおいて、当該障害の発生の検出を起点とした一連のステップから構成されるループを形成する当該プロセスにおいて、当該検出された障害発生、あるいは障害の予兆によるサービスの停止を回避する目的で前記ディペンダビリティ記述データを備えるシステムを再構成するステップと、当該再構成によるシステム運用とサービス提供によりステークホルダがアカウンタビリティを達成することができるステップという２つの処理を実行しても良い。

【０２２７】

また、本発明は以下のように構成してもよい。

【０２２８】

本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを仕様の変更に合わせて変更する変化対応サイクルＰ３００、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、

上記対象システムの停止を回避する障害対応サイクルＰ３０１のうち、少なくとも上記障害対応サイクルＰ３０１を実行するとともに、上記障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、当該対象システムを再構成する再構成手段（再構成部１０２－０４）を備えて構成されても良い。

【０２２９】

これにより、障害対応サイクルＰ３０１を実行するランタイムコンピュータ１０２において、対象システムの障害予兆を検知したとき、ディペンダビリティ記述データに基づいて、当該対象システムを再構成して、対象システムの運用を継続させることができる。

【０２３０】

そして、ランタイムコンピュータ１０２は、対象システムの障害予兆を、Ｄ値に基づいて検知してもよい。例えば、Ｄ値の時間変化の傾向が所定の基準より悪化したとき、障害予兆を検出するようにしても良い。

【０２３１】

さらに、ランタイムコンピュータ１０２は、対象システムの障害予兆を、前記モニタノードでのモニタリングデータに基づいて検知してもよい。例えば、当該モニタリングデータの時間変化の傾向が所定の基準より悪化し変動許容範囲を超えたとき、障害予兆を検出するようにしても良い。

【０２３２】

〔９．障害対応サイクルからの変化対応サイクルの起動〕

図２７は、ディペンダビリティ維持システム１００におけるディペンダビリティ記述データの変更抽出処理の手順を示すフローチャートである。

【０２３３】

上述したサービス継続判断Ｓ５０２－０１（図１９）、あるいは、サービス継続判断６０２－０２（図２３）において、サービス継続判断が「ＮＯ」すなわち継続不可能と判断されたときには、ディペンダビリティ記述データの変更抽出処理Ｓ８００を実行する。

【０２３４】

まず、障害部位に関するディペンダビリティ記述データを抽出する（Ｓ８００－０１）。次に、そのディペンダビリティ記述データにおけるリーフノードでテストと関連のあるノード全てにおいて、Ｓ８００－０３～Ｓ８００－０５の処理を実行する（Ｓ８００－０２）。

【０２３５】

例えば、図５においては、ノード群（２０２－０６～２０２－１３）が該当する。そこで、当該ノード群（２０２－０６～２０２－１３）に対応するテストを実行し計測する（Ｓ８００－０３）。既にエビデンスとしての結果は存在しているので、ここでの計測結果と比較することでエビデンスからの差異が変動許容範囲内か否かが判断される（Ｓ８００－０４）。変動許容範囲外の場合には（ＮＯ）、異常状態であるとして当該ノードの識別情報（例えば、ノード番号）を記録する（Ｓ８００－０５）。

【０２３６】

全ノードに対して前記処理を終えると、前記識別情報が記録されたノードを変更する指示をワークスペースコンピュータ１０１に送り（Ｓ８００－０６）、処理を終了する（Ｓ８００－０７）。

【０２３７】

なお、本発明は以下のように構成してもよい。

【０２３８】

本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、前記障害対応サイクルＰ３０１において、上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスの継続が不可能と判断された場合、前記変化対応サイクルＰ３００において前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、前記障害対応サイクルＰ３０１において記録又は検出された情報から前記ディペンダビリティ記述データを備えるシステムのディペンダビ

リティ記述データの変更を前記障害対応サイクルＰ３０１へ要求として入力するステップを実行しても良い。

【０２３９】

また、本発明は以下のように構成してもよい。

【０２４０】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ランタイムコンピュータ１０２）は、上記障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクルＰ３００を実行する装置（ワークスペースコンピュータ１０１）に、上記ディペンダビリティ記述データの変更の要求を送信する変更要求送信手段（変更送出部９０２－０２）を備えて構成されても良い。

【０２４１】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１）は、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを仕様の変更に合わせて変更する変化対応サイクルＰ３００、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルＰ３０１のうち、少なくとも上記変化対応サイクルＰ３００を実行するとともに、上記障害対応サイクルＰ３０１を実行する装置（ランタイムコンピュータ１０２）から、上記ディペンダビリティ記述データの変更の要求を受信したとき、上記ディペンダビリティ記述データを変更するように構成されても良い。

【０２４２】

これにより、障害対応サイクルＰ３０１を実行するランタイムコンピュータ１０２において、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であると判断したとき、変化対応サイクルＰ３００を実行するワークスペースコンピュータ１０１に、ディペンダビリティ記述データの変更の要求を送信することができる。なお、ランタイムコンピュータ１０２は、障害発生または障害予兆により、対象システムの停止が不回避であるか否かを、Ｄ値に基づいて判断しても良い。

【０２４３】

そして、ワークスペースコンピュータ１０１は、ランタイムコンピュータ１０２からディペンダビリティ記述データの変更の要求を受信したとき、当該要求に応じて、上記ディペンダビリティ記述データを変更する。

【０２４４】

したがって、ランタイムコンピュータ１０２とワークスペースコンピュータ１０１とが連携して、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であれば、ディペンダビリティ記述データを変更するというプロセスを円滑に実行することができる。よって、オープン環境において、対象システムのディペンダビリティを継続的に維持することが可能となる。

【０２４５】

なお、前記記載の構成において、対象システムの停止が回避可能である場合においても、障害予兆の内容によっては前記記載のディペンダビリティ記述データの変更手順を実行するように構成しても良い。

【０２４６】

〔１０．障害対応サイクルと変化対応サイクルとの連携〕

図２８は、ワークスペースコンピュータ１０１とランタイムコンピュータ１０２とのディペンダビリティ記述データを介した情報交換を示す機能ブロック図である。

【０２４７】

ワークスペースコンピュータ１０１における編集ツール部１０１－０１は、主に３個の構成要素を持つ。ディペンダビリティ記述データ編集部９００－０１は、ディペンダビリティ記述データを記述し、編集し、記録する。当該ディペンダビリティ記述データ編集部９００－０１は、図１６に記載のD-Case Editor（２６０－０６）として構成しても良い

。なお、ディペンダビリティ記述データ編集部９００－０１として、特願2010-263681に記載のツールを用いても良い。当該特願2010-263681では、要求ベースにディペンダビリティ記述データを記述する際に、常に着目ノードが表示の中心になるようにディペンダビリティ記述データを表示することで詳細化の流れを簡素にしている。

【０２４８】

ディペンダビリティ記述データ組込部９００－０２は、ディペンダビリティ記述データ編集部９００－０１にて作成されたディペンダビリティ記述データＭ１を内部表現に変換して、ディペンダビリティ記述データＭ２（正確には、ディペンダビリティ記述データ内部表現Ｍ２；図３のディペンダビリティ記述データ１０４－０２に相当）を生成する。そして、ディペンダビリティ記述データ組込部９００－０２は、ソフトウェアＳＷに、当該ソフトウェアＳＷに対応した（あるいは関する）前記ディペンダビリティ記述データＭ２を組み込む。デプロイメント部９００－０３は当該ソフトウェアＳＷをランタイムコンピュータ１０２に導入する。

【０２４９】

ここで、ディペンダビリティ記述データＭ２には、ディペンダビリティ記述データ編集部９００－０１によって、各エビデンスノードについて、モニタされるモニタ値に対する変動許容範囲が記録される。これにより、ランタイムコンピュータ１０２のモニタリング部１０２－０３では、モニタ値の状態を確認できる。一方、ディペンダビリティ記述データＭ３には、記録部１０２－０２によって、各エビデンスノードについて、モニタされたモニタ値のすべてあるいは一部が記録される。これにより、ワークスペースコンピュータ１０１の解析部９０１－０３では、システムに生じた異常の原因を検討することができる。

【０２５０】

また、ワークスペースコンピュータ１０１における解析ツール部１０１－０３は、主に３個の構成要素を持つ。ディペンダビリティ記述データ入力部（記述データ取得手段）９０１－０１はランタイムコンピュータ１０２からのディペンダビリティ記述データＭ３を入力とする。データベースアクセス部９０１－０２はランタイムコンピュータ１０２に導入されているソフトウェア全体に関する複数のディペンダビリティ記述データにアクセスする。データベースは、ワークスペースコンピュータ１０１に設けられても良いし、ランタイムコンピュータ１０２に設けられても良いし、ワークスペースコンピュータ１０１と通信可能な他の装置に設けられても良い。ここでは、入力されたディペンダビリティ記述データＭ３に関連したディペンダビリティ記述データＭ４を、当該データベースから抽出する。ディペンダビリティ記述データＭ４はオリジナルのモデルであり、ディペンダビリティ記述データＭ３は変更が記載されたモデルである。当該データベースから抽出されるディペンダビリティ記述データＭ４はディペンダビリティ記述データＭ３をキーに検索されるが、より広範囲のディペンダビリティ記述データを抽出しても良く、解析部９０１－０３での処理を容易にする。解析部９０１－０３は当該入力されたディペンダビリティ記述データＭ３とベータベースアクセス部９０１－０２経由で得られるディペンダビリティ記述データＭ４を解析して、障害原因等を分析し、必要に応じて環境変化Ｓ３００－０１（図１７、図１８）の状態に遷移させて、変化対応サイクルＰ３００を実行する。

【０２５１】

一方、ランタイムコンピュータ１０２において更新部１０２－０１は、６個の構成要素を持つ。ソフトウェア入力部（記述データ取得手段）９０２－０１はワークスペースコンピュータ１０１におけるデプロイメント部９００－０３からのソフトウェアＳＷを入力とする。当該ソフトウェアＳＷは更新部１０２－０１によってランタイムコンピュータ１０２にインストールされ稼働される。ディペンダビリティ記述データ抽出部９０２－０３はソフトウェアＳＷに含まれているディペンダビリティ記述データＭ２（正確には、ディペンダビリティ記述データ内部表現Ｍ２；図３のディペンダビリティ記述データ１０４－０２に相当）を取り出す。構造検証部９０２－０４は、当該取り出されたディペンダビリティ記述データＭ２の計算機表現としての妥当性を検証する。さらに、必要な機能（ライブ

ラリ等）がランタイムコンピュータ１０２に備わっているかを確認する。実行管理部９０２－０５は当該ソフトウェアＳＷをランタイムコンピュータ１０２に導入する際に必要な準備を行う。例えば、当該ソフトウェアＳＷが新しいサービスアプリケーションである場合には、必要な計算資源を見積もり割り当て、アプリケーション・コンテナ（図２５のアプリケーション・コンテナ７１７等）を生成し、アプリケーションイメージを当該アプリケーション・コンテナ内に構成する。

【０２５２】

また、変更送出部（変更要求送信手段）９０２－０２はランタイムコンピュータ１０２にインストールされたディペンダビリティ記述データＭ２における変更部分（ディペンダビリティ記述データＭ３）をワークスペースコンピュータ１０１のディペンダビリティ記述データ入力部９０１－０１に送り出す。このとき、変更送出部９０２－０２は、上記変更部分と共に、ディペンダビリティ記述データの変更要求を送出しても良い。変更管理部９０２－０６はランタイムコンピュータ１０２において、図２７に示したディペンダビリティ記述データの変更を管理する。

【０２５３】

前記において、ワークスペースコンピュータ１０１とランタイムコンピュータ１０２は、ディペンダビリティ記述データという計算機表現を介して情報交換する。前者から後者に対してはディペンダビリティ記述データＭ２を手段とし、後者から前者に対してはディペンダビリティ記述データＭ３を手段とする。ディペンダビリティ記述データＭ２はステークホルダ合意に対応したモデルであり、ディペンダビリティ記述データＭ３は当該ステークホルダ合意がランタイムコンピュータ１０２において満足されなくなり、その差分を表現したものであり、ステークホルダに対する通知である。

【０２５４】

なお、本発明は以下のように構成してもよい。

【０２５５】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、前記変化対応サイクルＰ３００と前記障害対応サイクルＰ３０１との間での情報交換の手段を有していても良い。

【０２５６】

〔１１．ディペンダビリティ記述データとの連携〕

図２９は、図５に示したディペンダビリティ記述データを、計算機表現の一例としてＸＭＬで記述したリストの一部である。なお、図２８におけるソフトウェアＳＷに組込まれたディペンダビリティ記述データＭ２、あるいはディペンダビリティ記述データＭ３は当該ＸＭＬ表記により記述することができる。

【０２５７】

図３０は、ディペンダビリティ維持システム１００の構成例であって、ディペンダビリティ記述データデータベースと、ワークスペースコンピュータおよびランタイムコンピュータとの関係を示す説明図である。

【０２５８】

図３０に示すように、ソフトウェアＳＷに組み込まれているディペンダビリティ記述データＭ２は、ワークスペースコンピュータ１０１およびランタイムコンピュータ１０２とは別体のディペンダビリティ記述データデータベース１０００に格納され、ワークスペースコンピュータ１０１およびランタイムコンピュータ１０２から利用されても良い。当該データベースは図１５に記載の合意記述データベース２５０－０６として構成しても良い。

【０２５９】

なお、本発明は以下のように構成してもよい。

【０２６０】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、前記ディペンダビリティ記述データから変換されたコンピュータで処理可能な形式のデータを、当該ディペンダビリティ記述データを備えるシステムが管理する手段を有することで、前記変化対応サイクルＰ３００におけるステークホルダによる変更の要求の合意のステップによる結果を、確実に当該ディペンダビリティ記述データを備えるシステムに実施する手段を有していてもよい。

【０２６１】

〔１２．ランタイムコンピュータでのコマンド処理〕

図３１に、ランタイムコンピュータ１０２におけるコマンド実行の処理手順を示す。本実施の形態におけるコマンドとは、あらかじめランタイムコンピュータ１０２に組み込まれた、当該ランタイムコンピュータ１０２で実行可能な一連の処理であっても良いし、後述のディペンダビリティ記述データのノードに関連付けられた当該ランタイムコンピュータ１０２で実行可能なプログラムであっても良い。

【０２６２】

まず、コマンド実行処理Ｓ１１００の開始にあたって、処理が開始された旨が記録される（Ｓ１１００－０１）。次に、処理を１個以上のコマンド列に分解し、それをチェインとして構成する（Ｓ１１００－０２）。つづいて、コマンドチェイン中の全コマンドに対して（Ｓ１１００－０３）、以下のＳ１１００－０４、Ｓ１１００－０５の処理が実行される。

【０２６３】

すなわち、チェイン中のコマンドを１つ実行し（Ｓ１１００－０４）、正常終了か、異常終了かを判断する（Ｓ１１００－０５）。そして、全部のコマンドが正常終了の場合には（Ｓ１１００－０５でＹＥＳ）、終了が記録され（Ｓ１１００－０７）、当該処理は終了する（Ｓ１１００－１４）。

【０２６４】

一方、コマンドチェイン中のコマンドが異常終了した場合（Ｓ１１００－０６でＮＯ）、実行されたコマンドの効果を取り消すためのリカバリチェインが構築される（Ｓ１１００－０８）。その後、リカバリチェイン中の全コマンドに対して（Ｓ１１００－０９）、リカバリコマンドを実行し（Ｓ１１００－１０）、正常終了か、異常終了かを判断する（Ｓ１１００－１１）。全リカバリコマンドが正常終了の場合（Ｓ１１００－１１でＹＥＳ）には、その旨を記録して（Ｓ１１００－０７）、当該処理は終了する（Ｓ１１００－１４）。

【０２６５】

一方、リカバリコマンドの実行が異常終了の場合には（Ｓ１１００－１１でＮＯ）、必要な隔離処理（Ｓ１１００－１３）を実行し、当該処理を含むコンテナ（アプリケーション・コンテナあるいはシステム・コンテナ）を隔離し、その旨を記録して（Ｓ１１００－０７）、当該処理は終了する（Ｓ１１００－１４）。その後、前記再構成処理を実行する。

【０２６６】

前記記載の手順は、図２４に記載の基本動作をコマンドチェインとするように構成しても良い。これにより、前記スクリプトをアトミック操作として確実に実行できるようになる。

【０２６７】

なお、本発明は以下のように構成してもよい。

【０２６８】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１、ランタイムコンピュータ１０２）は、前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、下記の〔Ａ〕〔Ｂ〕〔Ｃ〕に記載の手段群を、当該手段群が前記ディペンダビリティ記述データを備

えるシステムにおいて確実に実行されることを保証する手段を有していても良い。〔Ａ〕障害対応サイクルＰ３０１において、上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスの継続が不可能と判断された場合、変化対応サイクルＰ３００において前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、前記障害対応サイクルＰ３０１において記録又は検出された情報から前記ディペンダビリティ記述データを備えるシステムのディペンダビリティ記述データの変更を前記変化対応サイクルＰ３００に対する要求として当該変化対応サイクルＰ３００に入力する手段。〔Ｂ〕前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、変化対応サイクルＰ３００と障害対応サイクルＰ３０１との間での情報交換する手段。〔Ｃ〕ディペンダビリティ記述データから変換されたコンピュータで処理可能な形式のデータを、当該ディペンダビリティ記述データを備えるシステムが管理する手段を有することで、変化対応サイクルＰ３００におけるステークホルダによる変更の要求の合意による結果を、確実に当該ディペンダビリティ記述データを備えるシステムに実施する手段。

【０２６９】

〔１３．ノードへの実行可能プログラムの関連付け〕

図３２は、ディペンダビリティ記述データに関連付けられたプログラムの処理内容の一例を示すフローチャートである。

【０２７０】

図３２に示すノード（２０３－０４）は図６に示したものと同じノードである。当該ノードは「キャッシュによるApp Logicの継続」に関する。本実施の形態では、ディペンダビリティ記述データをランタイムコンピュータ１０２への組み込み手順に従って導入する。このとき、ディペンダビリティ記述データの各ノードには、ランタイムコンピュータ１０２のバイナリ実行部１１０Ｒで実行可能なプログラムを関連付けてもよい。また、モニタノードにも実行可能なプログラムが関連付けられて、ランタイムコンピュータ１０２に導入される。なお、ノードへのプログラムを関連付けは、外部参照情報１０４－０４に記述できる。また、これらの処理について図３２では当該プログラムの処理内容をフローチャートで表現したが、スクリプト言語を用いても良いし、バイナリ実行部１１０Ｒで直接実行可能なプログラムコードを用いても良い。

【０２７１】

プログラム１２００には導入処理Ｓ１２００－０１と廃止処理Ｓ１２００－１１の２個の処理の手順が定義されている。

【０２７２】

導入処理Ｓ１２００－０１は実行権限の確認（Ｓ１２００－０２）後に、アプリケーション・コンテナ提供部７１２（図２５）の機能を利用してアプリケーション・コンテナを生成（Ｓ１２００－０３）する。次に、App Logic２０１－０３（図４（ｂ））に対応するアプリケーションを読み込み、アプリケーション管理部７１４（図２５）に当該プログラムコードを登録する。

【０２７３】

一方、廃止処理Ｓ１２００－１１は実行権限の確認（Ｓ１２００－１２）後に、App Logic２０１－０３に対応するアプリケーションを停止（Ｓ１２００－１３）し、アプリケーション管理部７１４から当該プログラムコードの登録を削除し、アプリケーション・コンテナ提供部７１２の機能を利用して当該アプリケーション・コンテナを削除する。

【０２７４】

図３３は、ディペンダビリティ記述データに関連付けられたプログラムの処理内容の他の例を示すフローチャートである。

【０２７５】

図３３に示すノード（２０３－０１）は図７に示したものと同じノードである。当該ノードは「多重型によるＤＢＳの無停止」に関する。すなわち、バックアップ系を有するデータベースシステムを、複数台を同時に稼働させる多重系のデータベースに関するディペ

ンダビリティ記述データの一部であった。
【０２７６】
プログラム１２０１には更新処理Ｓ１２０１－０１、導入処理Ｓ１２０１－１１、廃止処理Ｓ１２０１－２１の３個の処理の手順が定義されている。ディペンダビリティ記述データにおけるノードが置き換えられるとき、当該ノードにプログラムが関連つけられている場合には、そこに定義された更新処理を実行する。
【０２７７】
具体的には、図３３では、ノード（２０３－０４）をノード（２０４－０１）で置き換えるので、ノードの更新処理が実行される。更新処理Ｓ１２０１－０１の手順は、実行権限の確認（Ｓ１２０１－０１）後に、置き換えられるノード（図３３の場合には図３２のノード（２０３－０４））に関連つけられた廃止処理Ｓ１２００－１１の手順を実行する。当該手順の実行に当たっては、図３１のコマンド実行処理Ｓ１１００のフローチャートに従い結果が記録（Ｓ１２０１－０４）される。
【０２７８】
次に、導入処理Ｓ１２０１－１１の手順は実行権限の確認（Ｓ１２０１－１２）後に、システム・コンテナ提供部７０２（図２５）の機能を利用してシステム・コンテナを作成する。次に、無停止ＤＢＳをインストール（Ｓ１２０１－１４）し、アプリケーション管理部７１４（図２５）に当該ＤＢＳを登録する。
【０２７９】
一方、廃止処理Ｓ１２０１－２１の手順は実行権限の確認（１２０１－２１）後に、無停止ＤＢＳを停止（Ｓ１２０１－２３）し、アプリケーション管理部７１４から当該ＤＢＳの登録を削除（Ｓ１２０１－２４）し、当該ＤＢＳのシステム・コンテナをシステム・コンテナ提供部７０２の機能を利用して削除（Ｓ１２０１－２５）する。
【０２８０】
なお、本発明は以下のように構成してもよい。
【０２８１】
本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、前記ディペンダビリティ記述データを備えるシステムが障害発生検出時および障害予兆検出時の障害対応サイクルＰ３０１を確実に実行するために、下記の手段群〔Ａ〕〔Ｂ〕をコンピュータで処理するための表現手段を有していても良い。〔Ａ〕上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスを障害発生に際しても継続するように対処するプロセスにおいて、障害の発生を検出する手段（モニタリング機能）と、障害の発生した部分を隔離する手段（隔離機能）と、隔離された残りの部分でサービスを継続する手段と、障害の原因を検出する手段（原因追及機能）と、上記システム運用時におけるシステムが提供するサービスの継続性を判断する手段（継続性判断機能）。〔Ｂ〕上記システム運用時における前記ディペンダビリティ記述データを備えるシステムが提供するサービスを停止させないように障害の予兆を検知し障害の発生を未然に回避するプロセスにおいて、前記ディペンダビリティ記述データを備えるシステムの内部状態を記録する手段（ロギング）と、当該記録された内部状態から前記ディペンダビリティ記述データを備えるシステムの障害の予兆を検出する手段と、前記ディペンダビリティ記述データを備えるシステムの当該検出された障害に対応する部分を同定する手段と、当該検出された障害が上記システム運用時におけるシステムが提供するサービスの継続性を判断する手段。
【０２８２】
また、本発明は以下のように構成してもよい。
【０２８３】
本発明のディペンダビリティ維持装置（ランタイムコンピュータ１０２）は、上記ディペンダビリティ記述データには上記再構成の手順が記載されており、上記再構成手段（再構成部１０２－０４）は、上記ディペンダビリティ記述データに記載されている再構成の上記手順に従って、当該対象システムを再構成するように構成されても良い。

【０２８４】
これにより、ランタイムコンピュータ１０２は、ディペンダビリティ記述データに記載されている再構成の手順に従って、当該対象システムを再構成することができる。よって、再構成の手順をディペンダビリティ記述データの設計段階において、当該ディペンダビリティ記述データに記述しておくことができるため、再構成時に手順を作成する必要がない。
【０２８５】
〔１４．合意形成ツールとの連携〕
図３４に、ディペンダビリティ記述データＭ１と、ソフトウェアＳＷに含まれるＸＭＬ形式ディペンダビリティ記述データＭ２（正確には、ディペンダビリティ記述データ内部表現Ｍ２であるが、ここでは、ＸＭＬ形式による内部表現を選択した場合について説明するので、「ＸＭＬ形式ディペンダビリティ記述データＭ２」と記す。）と、ワークスペースコンピュータ１０１が備えるツール群の各機能との関係を示す。
【０２８６】
編集機能１３００－０１（図１の編集ツール部１０１－０２が備える機能）は、ディペンダビリティ記述データＭ１を入力・修正・記録する。検証機能１３００－０２（図１の検証ツール部１０１－０１が備える機能）は、当該ディペンダビリティ記述データＭ１の妥当性を検証する。ベンチマーク機能１３００－０３（図１の検証ツール部１０１－０１が備える機能）は、当該ディペンダビリティ記述データＭ１におけるエビデンスを確認する。変換機能１３００－０４（図１の編集ツール部１０１－０２および開発ツール部１０１－０４が備える機能）は、差分検証モデルＭ１をソフトウェアＳＷへの組込形式である、ＸＭＬ形式ディペンダビリティ記述データＭ２に変換する。当該変換の際には、ディペンダビリティ記述データからモニタノード部分を抜き出し、その構造を維持しながら変換するように本実施例を構成しても良い。計測機能１３００－０５（図１の検証ツール部１０１－０１が備える機能）は、当該ＸＭＬ形式ディペンダビリティ記述データＭ２のモニタノードにアクセスしてデータを計測・抽出する。
【０２８７】
また、図３５にベンチマーク機能（図中ではDS-Benchとして記載）と図１６に記載のD-Case Editorとの連携を示す。D-Case Editor上ではモニタノードに実行すべきベンチマークを指定できるように構成している。当該D-Case Editorからのベンチマーク実行指示に基づいて実際のベンチマークが実行され、その実行結果を反映することでエビデンスとして記録する。当該ベンチマーク結果はＤ値の算出に用いることができる。また、Ｄ値に影響のあるベンチマークを用意することも可能であり、その場合、Ｄ値への影響が大きいほどベンチマークの有効性も高いことになる。
【０２８８】
当該ベンチマークには、システムの性能に関する特性以外の、障害の原因になるような特性を検証するベンチマークが含まれていてもよい。このようなベンチマークの例としては、ＣＰＵ負荷を変動させたときのシステムの振る舞いに関するベンチマーク、ネットワークの帯域幅を変動させたとき、または遅延を変動させたときのシステムの振る舞いに関するベンチマーク、余剰メモリを変動させたときのシステムの振る舞いに関するベンチマーク、過剰なアクセスをシミュレートすることでシステムに総合的に負荷をかけたときのシステムの振る舞いに関するベンチマーク、システムの構成コンポーネントを意図的に異常終了した際のシステムの振る舞いに関するベンチマーク、等が挙げられる。
【０２８９】
さらに、ディペンダビリティ記述データは、例えば図８に記載のD-Case記述を用いた場合、当該記述の整合性や網羅性等の検証のために、Agda定理証明器（http://wiki.portal.chalmers.se/agda/pmwiki.php）を用いるように構成しても良い。これにより、当該D-Case記述の記述エラーを自動的に取り除くことが可能になる。
【０２９０】
なお、本発明は以下のように構成してもよい。

【０２９１】
本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１）は、前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、前記変化対応サイクルＰ３００におけるステークホルダ間での合意を形成するステップにおいて、ステークホルダからの要求の変更をコンピュータで処理可能な表現形式で記述する手段を有し、当該表現形式の論理的検証手段を有していても良い。
【０２９２】
〔１５．ディペンダビリティ維持システム間の連携〕
本実施の形態に係るディペンダビリティ維持システム１００は、ワークスペースコンピュータ１０１、ディペンダビリティ記述データＭ１、ランタイムコンピュータ１０２から構成されると言える。そして、当該ディペンダビリティ維持システム１００の計算可能なディペンダビリティは、当該ディペンダビリティ記述データＭ１が表現している。
【０２９３】
ここで、図３６は、ディペンダビリティ維持システム１００を２つ接続した構成例を示すブロック図である。
【０２９４】
図３６に示すように、利用者のディペンダビリティ維持システム１００Ｕに対してディペンダビリティ記述データに基づく機能を提供する提供者のディペンダビリティ維持システム１００Ｓと、利用者の前記ディペンダビリティ維持システム１００Ｕとを、ネットワーク１４００－０３で接続した複合システムが構成できる。
【０２９５】
ここで、図３７は、図３６に示した２つのディペンダビリティ維持システム１００Ｕ，１００Ｓを接続した構成における、独立した２つのディペンダビリティ維持システム１００Ｕ，１００Ｓ間での連携の一例を示す説明図である。
【０２９６】
提供者のディペンダビリティ維持システム１００から、利用者のディペンダビリティ維持システム１００Ｕに対して、ＳＬＡ（Service Level Agreement）を含むＳＬＡディペンダビリティ記述データＭ２－ＳＬＡが提示される。ＳＬＡディペンダビリティ記述データＭ２－ＳＬＡは、ＳＬＡの情報を含む以外は、ディペンダビリティ記述データ内部表現Ｍ２と同じである。
【０２９７】
ＳＬＡディペンダビリティ記述データＭ２－ＳＬＡに含まれるＳＬＡには、ディペンダビリティ維持システム１００Ｓにて公開されるインターフェースのディペンダビリティに関してのみ記述される。これは提供者が提示する情報であり、当該インターフェースに対しての提供者のディペンダビリティに関してのコミットメントとして捉えることができる。また、当該ディペンダビリティ記述データはＤ値を計算可能であり、利用者はＤ値を計算して提供者の公開インターフェースに関してのディペンダビリティが利用者の基準に合致するかを確認できる。
【０２９８】
本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１）は、前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、２台以上の当該システムがネットワークによって相互接続された環境において、相互接続されていない環境において備えることが可能な上述した各手段を有していてもよい。
【０２９９】
〔１６．サプライチェーン〕
図３８は、図１に示したディペンダビリティ維持システム１００を２つ、本体側および部品側として統合した構成例を示すブロック図である。
【０３００】

図３８に示すように、機能の本体を実行する本体側のディペンダビリティ維持システム１００Ｂに対して、前記機能の一部を部品側のディペンダビリティ維持システム１００Ｐが部品として提供することが可能である。図３８では、ディペンダビリティ記述データＭ－Ｂにおけるノードの１個（１５００－０３）が部品としての提供されている例を示す。部品側のディペンダビリティ維持システム１００Ｐに示すディペンダビリティ記述データＭ－Ｐが、部品であるノード（１５００－０３）のディペンダビリティ記述データである。

【０３０１】

図３９は、２つのディペンダビリティ維持システム１００Ｂ，１００Ｐを本体側および部品側として統合した構成における、独立した２つのディペンダビリティ維持システム１００Ｂ，１００Ｐ間での連携の一例を示す説明図である。

【０３０２】

本体側のディペンダビリティ維持システム１００Ｂにおけるワークスペースコンピュータ１０１では、部品１５００－０３のディペンダビリティ記述データＭ－Ｐを検証する（Ｓ１５０２－０１）。当該部品１５００－０３のディペンダビリティ記述データＭ－Ｐが妥当である場合には、本体側のディペンダビリティ維持システム１００Ｂにおけるランタイムコンピュータ１０２で、当該部品１５００－０３のベンチマークを計測する（Ｓ１５０２－０２）。その結果のエビデンスが十分であれば、本体側のディペンダビリティ維持システム１００Ｂにおけるワークスペースコンピュータ１０１は当該部品１５００－０３を本体のディペンダビリティ記述データＭ－Ｂに統合する（Ｓ１５０２－０５）。

【０３０３】

図４０に、部品を統合する処理手順の詳細を示す。まず、当該部品１５００－０３のディペンダビリティ記述データＭ－Ｐを検証する（Ｓ１５０２－０１）。次に、ベンチマークを実行してエビデンスを計測し（Ｓ１５０２－０２）、さらに、Ｄ値を計算する（Ｓ１５０２－０３）。その結果、Ｄ値が良好である場合には、統合可能であり（Ｓ１５０２－０４でＹＥＳ）、統合のための再構成を実行する（Ｓ１５０２－０５）。一方、Ｄ値が満足できない場合には（Ｓ１５０２－０４でＮＯ）、部品提供者に対して部品（すなわち、そのディペンダビリティ記述データＭ－Ｐ）のアップデートを求める（Ｓ１５０２－０６）。以降、同様の手順を繰り返す。

【０３０４】

本発明のディペンダビリティ維持装置（ディペンダビリティ維持システム１００、ワークスペースコンピュータ１０１）は、前記ディペンダビリティ記述データを備えるシステムが提供するサービスを継続させる目的で、当該システムが第３者の１又は複数のハードウェア又はソフトウェア部品で構成される場合において、各々の部品が前記ディペンダビリティ記述データを備え、当該システムのディペンダビリティ記述データとの互換性の検証手段を有していても良い。

【０３０５】

〔１７．ディペンダビリティ記述データの表示〕

図４１を用いてランタイムコンピュータ１０２に格納されたディペンダビリティ記述データにおけるゴール群の一部が満足されなかった状況下で、ワークスペースコンピュータ１０１上の図１６に記載のD-Case Viewer（２６０－０７）が当該状況を表示する例を示す。ここでは、満足されなくなったゴールがフラッシュ（１６００－０１）されており、必要に応じてオペレータにアテンションを求める（１６００－０２）。なお、図４１はD-Case記述例であるが、記載内容は本説明では重要ではなく、その構造のみを示す。

【０３０６】

最後に、ディペンダビリティ維持システム１００の各ブロック、特にワークスペースコンピュータ１０１およびランタイムコンピュータ１０２の各ブロックは、ハードウェアロジックによって構成してもよいし、次のようにＣＰＵを用いてソフトウェアによって実現してもよい。

【０３０７】

後者の場合、ディペンダビリティ維持システム１００、あるいはワークスペースコンピュータ１０１およびランタイムコンピュータ１０２は、各機能を実現するプログラムの命令を実行するＣＰＵ（central processing unit）、上記プログラムを格納したＲＯＭ（read only memory）、上記プログラムを展開するＲＡＭ（random access memory）、上記プログラムおよび各種データを格納するメモリ等の記憶装置（記録媒体）などを備えている。そして、本発明の目的は、上述した機能を実現するソフトウェアであるディペンダビリティ維持システム１００、あるいはワークスペースコンピュータ１０１およびランタイムコンピュータ１０２の制御プログラムのプログラムコード（実行形式プログラム、中間コードプログラム、ソースプログラム）をコンピュータで読み取り可能に記録した記録媒体を、上記のディペンダビリティ維持システム１００、あるいはワークスペースコンピュータ１０１およびランタイムコンピュータ１０２に供給し、そのコンピュータ（またはＣＰＵやＭＰＵ）が記録媒体に記録されているプログラムコードを読み出し実行することによっても、達成可能である。

【０３０８】

上記記録媒体としては、例えば、磁気テープやカセットテープ等のテープ系、フロッピー（登録商標）ディスク／ハードディスク等の磁気ディスクやＣＤ－ＲＯＭ／ＭＯ／ＭＤ／ＤＶＤ／ＣＤ－Ｒ等の光ディスクを含むディスク系、ＩＣカード（メモリカードを含む）／光カード等のカード系、あるいはマスクＲＯＭ／ＥＰＲＯＭ／ＥＥＰＲＯＭ／フラッシュＲＯＭ等の半導体メモリ系などを用いることができる。

【０３０９】

また、上記のディペンダビリティ維持システム１００、あるいはワークスペースコンピュータ１０１およびランタイムコンピュータ１０２を通信ネットワークと接続可能に構成し、上記プログラムコードを通信ネットワークを介して供給してもよい。この通信ネットワークとしては、特に限定されず、例えば、インターネット、イントラネット、エキストラネット、ＬＡＮ、ＩＳＤＮ、ＶＡＮ、ＣＡＴＶ通信網、仮想専用網（virtual private network）、電話回線網、移動体通信網、衛星通信網等が利用可能である。また、通信ネットワークを構成する伝送媒体としては、特に限定されず、例えば、IEEE1394、ＵＳＢ、電力線搬送、ケーブルＴＶ回線、電話線、ＡＤＳＬ回線等の有線でも、ＩｒＤＡやリモコンのような赤外線、Bluetooth（登録商標）、802.11無線、ＨＤＲ、携帯電話網、衛星回線、地上波デジタル網等の無線でも利用可能である。なお、本発明は、上記プログラムコードが電子的な伝送で具現化された、搬送波に埋め込まれたコンピュータデータ信号の形態でも実現され得る。

【０３１０】

（１）本発明に係るディペンダビリティ維持装置は、対象システムのディペンダビリティを維持するためのディペンダビリティ維持装置であって、対象システムのディペンダビリティに関する（ステークホルダ群によって合意された）要求・仕様を記述したディペンダビリティ記述データを取得する記述データ取得手段と、上記記述データ取得手段が取得したディペンダビリティ記述データに基づいて、上記対象システムのディペンダビリティの価値を定量的に示す評価値を求めるディペンダビリティ値決定手段と、を備えることを特徴としている。

【０３１１】

（１２）また、本発明に係るディペンダビリティ維持装置の制御方法は、対象システムのディペンダビリティを維持するためのディペンダビリティ維持装置の制御方法であって、対象システムのディペンダビリティに関する（ステークホルダ群によって合意された）要求・仕様を記述したディペンダビリティ記述データを取得する記述データ取得ステップと、上記記述データ取得ステップにて取得したディペンダビリティ記述データに基づいて、上記対象システムのディペンダビリティの価値を定量的に示す評価値を求めるディペンダビリティ値決定ステップと、を含むことを特徴としている。

【０３１２】

上記の構成によれば、対象システムのディペンダビリティに関する（ステークホルダ群

によって合意された）要求・仕様を記述したディペンダビリティ記述データに基づいて、対象システムのディペンダビリティの価値を定量的に示す評価値を求めることができる。

【０３１３】

それゆえ、対象システムのディペンダビリティの価値を定量的に表現できる。したがって、例えば、要求変更に伴ってディペンダビリティ記述データを変更するときや、対象システムの運用時に対象システムの状態を確認するときに、対象システムのディペンダビリティの価値を分かりやすく客観的に提示することができる。

【０３１４】

よって、対象システムのディペンダビリティの維持を円滑に行うことが可能となるという効果を奏する。すなわち、不完全性と不確実性が潜在的に存在するオープン環境において、対象システムのディペンダビリティの維持を支援することが可能となるという効果を奏する。

【０３１５】

（２）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）に記載の構成において、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを（ステークホルダ群によって合意された）要求・仕様の変更に合わせて変更する変化対応サイクル、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルの少なくとも何れか一方を実行することを特徴としている。

【０３１６】

上記の構成によれば、さらに、変化対応サイクルおよび障害対応サイクルの少なくともいずれか一方において、対象システムのディペンダビリティの価値を示す評価値を求めることができる。したがって、例えば、変化対応サイクルにおいて、要求変更に伴ってディペンダビリティ記述データを変更するときに、評価値に基づいて、ディペンダビリティ記述データの変更案の適否を判断することが可能となる。また、例えば、障害対応サイクルにおいて、対象システムの運用時に対象システムの状態を、評価値に基づいて判断することが可能となる。よって、対象システムのディペンダビリティの維持を円滑に行うことが可能となるという効果を奏する。

【０３１７】

（３）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）に記載の構成において、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを（ステークホルダ群によって合意された）要求・仕様の変更に合わせて変更する変化対応サイクル、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルのうち、少なくとも上記障害対応サイクルを実行するとともに、上記障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、当該対象システムを再構成する再構成手段を備えることを特徴としている。

【０３１８】

上記の構成によれば、さらに、障害対応サイクルを実行するディペンダビリティ維持装置において、対象システムの障害発生あるいは障害予兆を検知したとき、ディペンダビリティ記述データに基づいて、当該対象システムを再構成して、対象システムの運用を継続させることができるという効果を奏する。なお、対象システムの障害発生または障害予兆を上記評価値に基づいて検知してもよい。例えば、上記評価値が所定の変動許容範囲より悪化したとき、あるいは、上記評価値の時間変化の傾向が所定の基準より悪化したとき、障害発生を検出するようにしても良い。

【０３１９】

（４）さらに、本発明に係るディペンダビリティ維持装置は、上記（３）に記載の構成において、上記ディペンダビリティ記述データには上記再構成の手順が記載されており、上記再構成手段は、上記ディペンダビリティ記述データに記載されている再構成の上記手順に従って、当該対象システムを再構成することを特徴としている。

【０３２０】
上記の構成によれば、さらに、ディペンダビリティ維持装置は、ディペンダビリティ記述データに記載されている再構成の手順に従って、当該対象システムを再構成することができる。よって、再構成の手順をディペンダビリティ記述データの設計段階において、当該ディペンダビリティ記述データに記述しておくことができるため、再構成時に手順を作成する必要がないという効果を奏する。
【０３２１】
（５）さらに、本発明に係るディペンダビリティ維持装置は、上記（３）または（４）に記載の構成において、上記障害発生または障害予兆を検知し、上記対象システムの停止が不回避であるとき、上記変化対応サイクルを実行する装置に、上記ディペンダビリティ記述データの変更の要求を送信する変更要求送信手段を備えることを特徴としている。
【０３２２】
上記の構成によれば、さらに、障害対応サイクルを実行するディペンダビリティ維持装置において、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であると判断したとき、変化対応サイクルを実行する装置に、ディペンダビリティ記述データの変更の要求を送信することができる。なお、障害発生または障害予兆により、対象システムの停止が不回避であるか否かを、上記評価値に基づいて判断しても良い。
【０３２３】
一方、変化対応サイクルを実行する装置は、上記ディペンダビリティ維持装置が送信したディペンダビリティ記述データの変更の要求を受信したとき、当該要求に応じて、上記ディペンダビリティ記述データを変更する。
【０３２４】
したがって、障害対応サイクルを実行するディペンダビリティ維持装置と、変化対応サイクルを実行する装置とが連携して、障害対応サイクルにおいて、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であれば、変化対応サイクルにおいて、ディペンダビリティ記述データを変更するという、一連のプロセスを円滑に実行することができる。よって、オープン環境において、対象システムのディペンダビリティを継続的に維持することが可能となるという効果を奏する。
【０３２５】
（６）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）に記載の構成において、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを（ステークホルダ群によって合意された）要求・仕様の変更に合わせて変更する変化対応サイクル、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルのうち、少なくとも上記変化対応サイクルを実行するとともに、上記障害対応サイクルを実行する装置から、上記ディペンダビリティ記述データの変更の要求を受信したとき、上記ディペンダビリティ記述データを変更することを特徴としている。
【０３２６】
上記の構成によれば、さらに、変化対応サイクルを実行するディペンダビリティ維持装置において、障害対応サイクルを実行する装置から、ディペンダビリティ記述データの変更の要求を受信したとき、当該要求に応じて、上記ディペンダビリティ記述データを変更することができる。
【０３２７】
したがって、障害対応サイクルを実行する装置と、変化対応サイクルを実行するディペンダビリティ維持装置とが連携して、障害対応サイクルにおいて、対象システムの障害発生または障害予兆を検知し、対象システムの停止が不回避であれば、変化対応サイクルにおいて、ディペンダビリティ記述データを変更するという、一連のプロセスを円滑に実行することができる。よって、オープン環境において、対象システムのディペンダビリティを継続的に維持することが可能となるという効果を奏する。

【０３２８】
（７）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）から（６）のいずれかに記載の構成において、上記ディペンダビリティ記述データは、互いに関連づけられたゴールノードとモニタノードとの組を規定するデータであり、上記ゴールノードは、（ステークホルダ群によって合意された）要求・仕様がゴール形式にて記述されたノードであり、上記モニタノードは、上記ゴールノードに記述されたゴールが充足されていることを表明するエビデンスであるとともに、上記対象システムの監視点に対応付けられており、上記ディペンダビリティ値決定手段は、対応付けられた上記監視点から取得されたデータが変動許容範囲に対して良好であるモニタノードに基づいて、上記評価値を計算することを特徴としている。
【０３２９】
上記の構成によれば、さらに、互いに関連づけられたゴールノードとモニタノードとの組を規定するデータ、例えば、木構造あるいはグラフ構造のデータから、多次元ベクトル値として評価値を求めることができる。
【０３３０】
また、本発明に係るディペンダビリティ維持装置は、上記（１）から（６）のいずれかに記載の構成において、
・上記ディペンダビリティ記述データは、ゴールノードとモニタノードとを組にすることが可能であるとともに、当該組を１以上有する木構造あるいはグラフ構造を有するデータであって、
・上記ゴールノードは、（ステークホルダ群によって合意された）要求・仕様がゴール形式にて記述することが可能なノードであり、
・上記モニタノードは、上記ゴールノードに記載のゴールが満たされていることを表明するエビデンスであり、
さらに、上記モニタノードは、上記対象システム内部の対応した監視点からのデータ取得を可能にした上で、当該監視点データが変動許容範囲に対して良好であることを判断する手段を備えており、
上記ディペンダビリティ値決定手段は、上記エビデンスが良好であるモニタノードを上記木構造あるいはグラフ構造において計算することで上記評価値とすることを特徴としている。
【０３３１】
また、上記ディペンダビリティ維持装置は、上記ディペンダビリティ記述データがノード間のエッジに重み付けが可能な木構造あるいはグラフ構造を有するデータであり、上記ディペンダビリティ値決定手段は、上記エッジに重み付けされた重みを要素とする多次元ベクトル値を上記評価値としてもよい。
【０３３２】
上記の構成によれば、さらに、木構造あるいはグラフ構造のエッジに重み付けされた重みを要素とする多次元ベクトル値を評価値として求めることができる。
【０３３３】
それゆえ、木構造あるいはグラフ構造に、エッジの重要性を加味した評価値が得られる。よって、評価値を確認することで、エッジに重み付けの変更がディペンダビリティに与える影響を評価することができるという効果を奏する。なお、ここで得られる評価値は多次元ベクトル値であるが、目的に応じた任意の変換手法によりスカラー値に変換できることはいうまでもない。
【０３３４】
（８）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）から（６）のいずれかに記載の構成において、上記ディペンダビリティ値決定手段は、上記対象システムをモニタして取得されたモニタ値が変動許容範囲に対して良好であったエビデンスに基づいて、上記評価値を計算することを特徴としている。
【０３３５】

上記の構成によれば、さらに、対象システムをモニタして取得されたモニタ値が変動許容範囲（基準値）に対して良好であったエビデンスに基づいて、評価値を計算することができる。その計算方法としては、例えば、ディペンダビリティ記述データに含まれるエビデンスの総数（総エビデンス数）に対する、モニタ値が変動許容範囲（基準値）に対して良好なエビデンスの数（有効エビデンス数）の割合を評価値として計算することができる。

【０３３６】

それゆえ、エビデンスの状態に応じた評価値が得られる。エビデンスは要求変更に応じて変化するため、ここで得られる評価値も要求変更に応じて変化する。すなわち、評価値を確認することで、要求変更がディペンダビリティに与える影響を評価することができるという効果を奏する。

【０３３７】

（９）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）に記載の構成において、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを（ステークホルダ群によって合意された）要求・仕様の変更に合わせて変更する変化対応サイクル、および、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルのうち、少なくとも上記障害対応サイクルを実行するとともに、上記障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに含まれるスクリプトを実行するスクリプト処理手段を備え、上記スクリプトは上記対象システムを変動許容範囲の状態に回復させるシナリオを含むことを特徴としている。

【０３３８】

上記の構成によれば、さらに、障害発生または障害予兆を検知したとき、ディペンダビリティ記述データに含まれるスクリプトを実行することによって、対象システムを変動許容範囲の状態に回復させることができる。

【０３３９】

なお、上記スクリプトは、ステークホルダによって合意されていることが好ましい。また、上記スクリプトは、対象システムに組み込まれたスクリプトエンジンによって実行される。また、上記スクリプトは、対象システム内部で取得されているログから、対象システムの状態が変動許容範囲にあるかを判断してもよい。また、対象システムを変動許容範囲の状態に回復させるシナリオを実行する際、当該シナリオは、ＧＵＩを介してオペレータから操作を受け付けてもよい。

【０３４０】

（１０）また、本発明に係るディペンダビリティ維持システムは、対象システムのディペンダビリティを維持するためのディペンダビリティ維持システムであって、上記対象システムの開発時あるいは更新時に、上記ディペンダビリティ記述データを（ステークホルダ群によって合意された）要求・仕様の変更に合わせて変更する変化対応サイクルを実行するワークスペース装置と、上記対象システムの運用時に、障害発生または障害予兆を検知したとき、上記ディペンダビリティ記述データに基づいて、上記対象システムの停止を回避する障害対応サイクルを実行するランタイム装置とを含み、上記ワークスペース装置および上記ランタイム装置の少なくともいずれか一方が、上記ディペンダビリティ記述データに基づいて、上記対象システムのディペンダビリティの価値を定量的に示す評価値を求めることを特徴としている。

【０３４１】

（１１）さらに、本発明に係るディペンダビリティ維持システムは、上記（１０）に記載の構成において、他のディペンダビリティ維持システムとネットワークを介して接続されていることを特徴としている。

【０３４２】

なお、上記のディペンダビリティ維持装置は、コンピュータによって実現してもよく、この場合には、コンピュータを上記の各手段として動作させることにより上記ディペンダ

ビリティ維持装置をコンピュータにて実現させるディペンダビリティ維持装置のプログラム、およびそれを記録したコンピュータ読み取り可能な記録媒体も、本発明の範疇に入る。

【０３４３】

〔実施の形態２〕

本実施の形態に係るディペンダビリティ維持装置２０は、要求・仕様書およびシステムの更新の間の無矛盾性を維持することを容易にするものである。そのため、後述するように、本実施の形態に係るディペンダビリティ維持装置２０は、監視制御対象システム２のディペンダビリティに関する仕様を記述したD-CaseをD-Case格納部１０から読み出し、監視制御対象システム２の状態を監視し、必要な場合に対策を実行する障害監視制御部３０の動作を制御する監視制御データを、上記読み出したD-Caseから生成し、障害監視制御部３０に供給するD-Case変換部２１を備える。

【０３４４】

以下、図４２から図５６、および図８に基づいて、本発明の一実施の形態について詳細に説明する。

【０３４５】

図４２は、本実施の形態に係る障害対応システム１の構成の概略を示す機能ブロック図である。

【０３４６】

本実施の形態に係る障害対応システム１は、監視制御対象システム２の動作を監視し、障害を発生した場合に、必要な対応措置を行うものである。なお、監視制御対象システム２としては、任意のコンピュータシステムに適用可能であるが、特に、単一のシステムではなく、複数のステークホルダが存在する、高度で複雑なディペンダビリティが必要とされているシステムに好適である。具体的には、社会インフラシステムの基盤であるようなコンピュータシステム、例えば、監視システム、電子決済システム、交通・航空管制システム、さらには、それらを包含するクラウドシステムに好適である。

【０３４７】

そして、特に、本実施の形態に係る障害対応システム１は、監視制御対象システム２の各ステークホルダの合意の記述を容易にするとともに、監視制御対象システム２の一部更新があった場合に、合意の記述と監視制御のための実装モジュールの間が無矛盾に更新されるように維持することを特徴とするものである。すなわち、障害対応システム１は、ディペンダビリティ記述データで表現された監視制御対象システム２のディペンダビリティに関わる仕様と、監視制御対象システム２を監視制御するためのモニタモジュールやアクションモジュールの制御との間を、無矛盾に開発・更新することを可能とする。

【０３４８】

そのために、障害対応システム１は、ディペンダビリティ記述データとして、各ステークホルダの合意の記述を容易に行うことができるという特徴を持つD-Caseを用いて、監視制御対象システム２のディペンダビリティを記述する。また、障害対応システム１は、D-Caseのパターン（D-Caseパターン）と監視制御対象システム２を監視制御するためのモニタモジュールおよびアクションモジュールとの対応を管理する。これにより、監視制御対象システム２の変更等に応じて、モニタモジュールやアクションモジュールが追加・変更されても、ディペンダビリティ記述とモニタモジュールおよびアクションモジュールとの対応を維持することができる。

【０３４９】

（D-Case）

図８は、D-Caseの基本構造を示す説明図である。また、図４３は、D-Caseの具体例を示す説明図である。なお、図４３は、顔認識を用いた監視システムを監視制御対象システム２とした場合のD-Caseである。

【０３５０】

上記の通り、本実施の形態では、監視制御対象システム２のディペンダビリティに関す

る仕様を記述するディペンダビリティ記述データとして、D-Caseを使用する。

【０３５１】

D-Caseとは、Safety Caseという、主にイギリスで用いられている、システムの安全性を保証するために用いられる構造化されたドキュメントを元にしている。本明細書では、Safety Caseを元にしたディペンダビリティ記述データを「D-Case」と称するものとする。D-Caseによれば、可用性などの一般的なディペンダビリティ属性は、図４３のようなシステムレベルの議論に詳細化される。

【０３５２】

D-Caseは、監視制御対象システム２に関わる各ステークホルダが合意するための構造化されたドキュメントである。詳細には、図８に示すように、D-Caseは、システムのディペンダビリティをトップゴールとして、それが満たされていることを木構造により詳細化していき、詳細化されたゴールに対してエビデンスを置いた木構造を有する。D-Caseは、後述するようにＸＭＬ（Extensible Markup Language）によって記述することができる。

【０３５３】

ここで、図４３中、“Ｇ”を付したブロックがトップゴールである。トップゴールとは、対象システムに対して、ステークホルダ間で合意すべき命題を表す。例えば「対象システムは、安全性規格Xで定義されているSafety Integrity Level 3を満たす」などである。

【０３５４】

“ＳＧ”を付したブロックがサブゴールである。サブゴールとは、トップゴールを示すために、示すべきことを分割した命題を表す。例えば、トップゴールが「Ａである、かつ、Ｂである」であるとき、サブゴール「Ａである」、「Ｂである」に分割する。サブゴールは、さらに小さなサブゴールに分割されうる。トップゴール、サブゴールを総称して、ゴールと呼ぶ。

【０３５５】

“Ｓｔ”を付したブロックがストラテジである。ストラテジとは、ゴールが成り立つことを、そのサブゴール群によってどのように議論されるのかの補助説明を示す。例えば、ゴールが「システムは考えられる障害すべてに対応する」であるとき、ストラテジとして、「考えられる障害ごとに議論する」をあげる。この場合、サブゴールは、「システムは考えられる障害１に対応する」、…、「システムは考えられる障害Ｎに対応する」となる。

【０３５６】

“Ｅ”を付したブロックがエビデンスである。エビデンスとは、分割され、詳細化されたゴールを、最終的に保証するリーフノードである。例えば、サブゴールＳＧ「システムＸのサブコンポーネントＹは、障害Ｚに対処できる」というゴールに対して、「ＦＴＡ解析結果」というエビデンスをサブゴールＳＧの直下におく。サブゴールＳＧをさらにサブゴールに分割することも考えられるが、無限にゴールを分割することはできない。分割されたサブゴールを最終的に保証するものとして、エビデンスがおかれる。エビデンスの妥当性は，ステークホルダ間の合意に基づく。

【０３５７】

また、図４７に示されるように、“Ｃ”を付したブロックがコンテキストである。コンテキストは、ゴールやストラテジの内容を補足する情報を示す。例えば、ストラテジ「考えられる障害ごとに議論する」である場合、コンテキストとして「考えられる障害のリスト」がそのストラテジに付けられうる。ストラテジやコンテキストは、D-Caseを読むステークホルダが、ゴールの分割を追いながら読むときの助けとなる補助情報を示す。

【０３５８】

なお、ストラテジ、コンテキストは、木構造化するときには読みやすさのために記入するが、ランタイム時にはコメント扱いである。

【０３５９】

また、図５０に示される、“Monitor:M_1”等のブロックがモニタである。モニタとは

、システムが障害対応を行う際に必要になるランタイム時のシステムの情報を表す。例えば、ランタイム時の「プロセスＸのＣＰＵ使用率のモニタリング」などがモニタノードとなる。モニタはエビデンスのサブクラスである。障害対応システム１のモニタ管理部４０により得られる情報をエビデンスとして用いる場合に、そのエビデンスをモニタノードとして表す。

【０３６０】

つづいて、図４２および図４４を参照しながら、上記障害対応システム１の構成および処理フローについて詳細に説明する。

【０３６１】

図４２に示すように、障害対応システム１は、D-Case格納部（ディペンダビリティ記述格納部）１０、ディペンダビリティ維持装置２０、障害監視制御部３０、モニタ管理部４０、モニタモジュール群５０、アクション管理部６０、アクションモジュール群７０を備えて構成されている。

【０３６２】

上記D-Case格納部１０は、監視制御対象システム２についての、ステークホルダ間のディペンダビリティに関する合意を記述したディペンダビリティ記述データであるD-Caseを格納する。なお、障害対応システム１では、各ステークホルダが合意したD-Caseが入力となる。

【０３６３】

上記ディペンダビリティ維持装置２０は、監視制御対象システム２を監視制御するための実装モジュール（モニタモジュール、アクションモジュール）の制御とD-Caseパターンとの対応を示すD-Caseパターン<=>モジュール対応テーブル（後述する）を使用して、D-Caseから監視制御データ（障害対応スクリプト）を生成する。そのために、ディペンダビリティ維持装置２０は、D-Case変換部（ディペンダビリティ記述変換部）２１、対応テーブル格納部２２を備えて構成されている。

【０３６４】

上記D-Case変換部２１は、障害監視制御部３０の動作を制御する監視制御データ（障害監視制御用データ）を、D-Case格納部１０に格納されているD-Caseから生成する。

【０３６５】

ここで、D-Caseは、その一部分がパターン化されている。そして、上記対応テーブル格納部２２は、D-Caseパターンと監視制御データとの対応を表わすテーブル（D-Caseパターン<=>モジュール対応テーブル）を格納している。

【０３６６】

D-Case変換部２１は、対応テーブル格納部２２に格納されているD-Caseパターン<=>モジュール対応テーブルを参照して、D-Caseから監視制御データを生成する。

【０３６７】

つぎに、上記障害監視制御部３０は、監視制御対象システム２の状態を監視（モニタ）し、必要な場合に対策（アクション）を実行する。具体的には、障害監視制御部３０は、ディペンダビリティ維持装置２０においてD-Case変換部２１が生成した監視制御データに従って、モニタ管理部４０とアクション管理部６０を制御する。監視制御データは、モニタ管理部４０によるモニタモジュール（モニタモジュール群５０）の選択および動作の制御、並びに、アクション管理部６０によるアクションモジュール（アクションモジュール群７０）の選択および制御を規定している。

【０３６８】

上記モニタ管理部４０は、監視制御データに従って、１以上のモニタモジュール（モニタモジュール群５０）を管理する。本実施の形態では、モニタモジュールの例として、ＣＰＵ監視モジュール５１、メモリ監視モジュール５２を挙げるが、本発明はこれに限定されない。

【０３６９】

上記アクション管理部６０は、監視制御データに従って、１以上のアクションモジュー

ル（アクションモジュール群７０）を管理する。本実施の形態では、アクションモジュールの例として、ＣＰＵ制限モジュール７１、メモリ制限モジュール７２、プロセス死活モジュール７３を挙げるが、本発明はこれに限定されない。

【０３７０】

図４４は、障害対応システム１の処理を示すフローチャートである。

【０３７１】

まず、D-Case管理者が、ステークホルダ間の合意のもとで作成、変更されたD-Caseを、D-Case格納部１０に格納する（Ｓ１；D-Case格納ステップ）。

【０３７２】

次に、D-Case変換部２１が、D-Case格納部１０からD-Caseを読み出し（Ｓ２；D-Case読み出しステップ）、読み出したD-Caseから、対応テーブル格納部２２に格納されているD-Caseパターン<=>モジュール対応テーブルを参照して、障害監視制御部３０の動作を制御する監視制御データを生成する（Ｓ３；D-Case変換ステップ）。

【０３７３】

なお、D-Caseの変更に伴って監視制御データを生成する場合には、D-Case変換部２１は、変更されたD-Caseの差分のみに対応する監視制御データを作成してもよい。

【０３７４】

次に、障害監視制御部３０が、作成された監視制御データに従って、監視制御対象システム２の状態を監視（モニタ）し、必要な場合に対策（アクション）を実行する（Ｓ４；障害監視実行ステップ）。

【０３７５】

ここで、上記障害監視実行ステップ（Ｓ４）について説明する。

【０３７６】

まず、モニタ管理部４０には、管理すべきモニタモジュールがあらかじめ制御可能に登録されている。また同様に、アクション管理部６０には、管理すべきアクションモジュールがあらかじめ制御可能に登録されている。

【０３７７】

そして、障害監視実行ステップ（Ｓ４）では、障害監視制御部３０が、監視制御データに従ってモニタ管理部４０に適切なモニタモジュールを起動するよう指示する。そして、障害監視制御部３０は、モニタ管理部４０から通知されたモニタモジュールの実行結果が条件を満たす場合、さらに、アクション管理部６０に適切なアクションモジュールを起動して障害に対処するように指示する。このとき、モニタ管理部４０およびアクション管理部６０は、それぞれ、障害監視制御部３０からの指示に従ってモジュールを起動し、適切な引数を渡す。

【０３７８】

（D-Case変換）

つぎに、図４５～図４９を参照しながら、D-Caseから監視制御データへの変換処理について詳細に説明する。

【０３７９】

障害対応システム１では、D-Caseから、監視制御対象システム２を監視制御するためのモニタモジュールおよびアクションモジュールの動作を制御する監視制御データへの変換に、D-Caseパターン<=>モジュール対応テーブルを用いる。

【０３８０】

ここで、D-Caseパターンとは、監視制御対象システム２のD-Caseの一部であり、可用性などが、システムレベルでどのように維持されるかを示すものである。すなわち、システムの可用性とは、システムが障害などでサービスを行えない状態になったとき（システムダウン状態）、できるだけ早急に復旧し、利用者がサービスを用いたい時にいつでも用いられるようにできるシステムの性質である。障害が発生した場合、保守管理者などの人員により迅速に行うこと、およびシステム自体が自動的に障害復旧機能により復旧を行う。一般にシステムは、ＯＳレベルにおいては、ＣＰＵ資源、メモリ資源、ネットワーク資源

などを用いて、サービスを提供する。したがって、ＣＰＵ資源などが不足した場合、サービスの遅延などが発生し、可用性が減じる可能性がある。よって、あるサービスに用いるＣＰＵ資源が減じた場合、他の重要度の低いサービスで用いられているＣＰＵ資源を転用するなどの対応をシステムはＯＳレベルで自動的に行う。監視制御対象システム２のD-Caseは、上記観点から、システムレベルでの可用性について議論している。

【０３８１】

（D-Caseパターン）

図４５は、D-Caseパターンを含むD-Caseの具体例を示す説明図である。

【０３８２】

D-Caseパターンとは、D-Caseの部分木であり、可変な部分を含むものである。図４５はパターンの一例であり、プロセスの属性が適切に保たれることを表現する場合に使用される。サブゴールＳＧ２２とエビデンスＥ２５、Ｅ２６の「」の部分が可変（変数）であり、他は固定されたD-Caseパターンである。図４５は、「」内に既に「画像処理」、「ＣＰＵ使用率」、「５０％以下」を代入した状態を示している。このD-CaseパターンがD-Caseの一部として使用される場合に、「」部分にはステークホルダ間の合意に基づく値が代入され、その代入される値により、モニタモジュールやアクションモジュールと適切に関連づけることが可能となる。また、他は固定されたD-Caseである。D-Caseパターン、すなわちD-Caseの「」部分には、ステークホルダ間の合意に基づく値が代入される。また、D-Caseの「」部分に値が代入されることにより、モニタモジュールやアクションモジュールと適切に関連づけることが可能となる。

【０３８３】

図４６は、D-Caseパターン<=>モジュール対応テーブルの一例であって、（ａ）はモニタモジュールについての対応テーブルを示し、（ｂ）はアクションモジュールについての対応テーブルを示す。

【０３８４】

図４６（ａ）は、≪プロセス「$1」の「$2」が「$3」であることをモニタする≫というD-Caseを、D-Case変換部２１が変換するときに参照するテーブルである。このD-Caseでは、$1、$2、$3が変数である。

【０３８５】

例えば、図４５のD-Caseを処理する際、D-Case変換部２１は、エビデンスＥ２５の記述≪プロセス「画像処理」の「ＣＰＵ仕様率」が「５０％以下」であることをモニタする≫に当てはめることができるテーブル（図４６（ａ））を対応テーブル格納部２２から読み出す。そして、エビデンスＥ２５の記述から抽出した変数$1、$2、$3の値を用いて、D-Caseパターンとモジュールとの対応関係を特定し、監視制御データを作成する。具体的には、変数$2が“ＣＰＵ仕様率”であることから、モジュール名“ＣＰＵ監視”および引数“$1、$3”を特定し、≪「ＣＰＵ監視」モジュールを呼び出し、「画像処理」「５０％以下」を引数として渡す≫という内容の監視制御データを作成する。

【０３８６】

同様に、D-Case変換部２１は、エビデンスＥ２６の記述≪プロセス「画像処理」の「ＣＰＵ仕様率」を「５０％以下」に制限する≫に当てはめることができるテーブル（図４６（ｂ））を対応テーブル格納部２２から読み出す。そして、変数$2が“ＣＰＵ仕様率”であることから、モジュール名“ＣＰＵ制限”および引数“$1、$3”を特定し、≪「ＣＰＵ制限」モジュールを呼び出し、「画像処理」「５０％以下」を引数として渡す≫という内容の監視制御データを作成する。

【０３８７】

上記のように、エビデンスの記述のうち、固定されたD-Caseの文字列（記述の固定部分）に応じて、D-Caseパターン<=>モジュール対応テーブルが選択される。そして、一部の変数（上記の例では、変数$2）の値に応じて、対応するモジュールが選択される。

【０３８８】

このように、本実施の形態では、エビデンスの記述を一部に変数が設定された表現形式

とし、そのエビデンスに則した値を変数に設定して、エビデンスを記述する。

【０３８９】

つぎに、図４７、図４８を参照して、図４５のD-Caseが変更された場合の処理について説明する。

【０３９０】

図４７は、D-Caseパターンを含むD-Caseの具体例を示す説明図である。

【０３９１】

図４７では、図４５に対して、コンテキストＣ３１が追加されるとともに、エビデンスＥ２６がエビデンスＥ３２に変更されている。これは、≪Ｃ３１：他のプロセス「顔認識」を制限してでも「画像処理」にＣＰＵを十分与えたい≫ため、≪Ｅ３２：プロセス「顔認識」を「kill」する≫ことで、各ステークホルダが合意したことを示している。

【０３９２】

図４８は、このときのD-Caseパターン<=>モジュール対応テーブルの一例である。図４８（ａ）はモニタモジュールについての対応テーブルを示し、図４８（ｂ）はアクションモジュールについての対応テーブルを示す。図４８（ａ）は、図４６（ａ）と同じである。

【０３９３】

そして、図４７のD-Caseを処理する際、D-Case変換部２１は、エビデンスＥ３２の記述≪プロセス「顔認識」を「kill」する≫に当てはめることができるテーブル（図４８（ｂ））を対応テーブル格納部２２から読み出す。そして、エビデンスＥ３２の記述から抽出した変数$1、$2の値を用いて、D-Caseパターンとモジュールとの対応関係を特定し、監視制御データを作成する。具体的には、変数$2が“Kill”であることから、モジュール名“プロセス死活”および引数“$1”を特定し、≪「プロセス死活」モジュールを呼び出し、「Kill」「顔認識」を引数として渡す≫という内容の監視制御データを作成する。

【０３９４】

このように、D-Caseパターン<=>モジュール対応テーブルには、引数として、変数だけでなく、定数値を設定しておくこともできる。すなわち、図４８（ｂ）では、変数$2の値が“Kill”であれば、プロセス死活モジュール７３へ渡す１番目の引数は“Kill”となり、変数$2の値が“Restart”であれば、渡す引数は“Restart”となる。また、変数$2の値が“Migration”であれば、プロセス死活モジュール７３ではなく、Migrationモジュール（図示せず）がアクションとして実行されることになる。

【０３９５】

つぎに、図４９を参照して、新たにモニタモジュールやアクションモジュールを追加した場合の処理について説明する。

【０３９６】

新たにモニタモジュールやアクションモジュールを追加した場合には、モニタ管理部４０やアクション管理部６０に登録する前に、D-Case管理者が、D-Caseパターン<=>モジュール対応テーブルに対応関係を規定するデータ（変換規則）を追加する。

【０３９７】

図４９は、D-Caseパターン<=>モジュール対応テーブルの一例であって、モニタモジュールについての対応テーブルを示す。例えば、ネットワーク転送量を監視するモジュールを新たに追加した場合、図４９のように、図４６（ａ）に１行追加される。

【０３９８】

（まとめ）

以上のように、上記障害対応システム１では、モニタ管理部４０によるモニタモジュールの選択および動作の制御、並びに、アクション管理部６０によるアクションモジュールの選択および制御と、D-Caseパターンとの対応を示すD-Caseパターン<=>モジュール対応テーブルを使用して、D-Caseから監視制御データを生成する。すなわち、D-Case格納部１０に格納されたD-Caseは、D-Caseパターン<=>モジュール対応テーブルを参照しながら、障害監視制御部３０の監視制御データに変換される。このように、障害監視制御部３０が

用いる監視制御データが、D-Case部１０に格納されたD-Caseから、D-Case変換部２１によって生成されることにより、D-Caseと障害監視制御部３０の動作との間が常に無矛盾になるように維持される。

【０３９９】

また、D-Case格納部１０に格納されているD-Caseを変更した場合、その変更はD-Case変換部２１によって監視制御データの変更として障害監視制御部３０に通知される。なお、D-Caseの変更をディペンダビリティ維持装置２０あるいはD-Case変換部２１が検知し、変更後の監視制御データを障害監視制御部３０に自動的に通知するようにしてもよい。

【０４００】

また、D-Case管理者は、D-Caseに新たにD-Caseパターンを追加した場合、D-Caseパターン<=>モジュール対応テーブルに対応関係を規定するデータ（変換規則）を追加する。

【０４０１】

また、モニタモジュールやアクションモジュールを修正・追加した場合、障害対応システム１はD-Case管理者に通知する。D-Case管理者は、修正・追加されたモニタモジュールやアクションモジュールに対応するD-Caseパターン<=>モジュール対応テーブルも合わせて修正する。なお、修正・追加されたモニタモジュールやアクションモジュールに対応するD-Caseパターン<=>モジュール対応テーブルが修正されない場合、モニタモジュールあるいはアクションモジュールがD-Caseと対応できてないので、監視制御データが使用できない。修正されたD-Caseパターン<=>モジュール対応テーブルを使用して監視制御データを修正することによって、D-Caseと障害監視との間の無矛盾性が維持される。

【０４０２】

また、D-Caseパターン<=>モジュール対応テーブルは、データベースに蓄積しておき、D-Caseを変更する際、適切なD-Caseパターン<=>モジュール対応テーブルを参照するように、データベースから抽出して利用するようにしてもよい。これにより、D-Caseの変更の度にD-Caseパターン<=>モジュール対応テーブルを作成する必要がなくなり、低コストでD-Caseの変更に対応することが可能となる。

【０４０３】

つづいて、図５０～図５６を参照しながら、他の具体例について説明する。

【０４０４】

図５０および図５１は、D-Caseの他の具体例を示す説明図である。図５０と図５１は連結部Ａで連結されて、一つの木構造となる。また、図５２～図５５は、図５０、図５１に示したD-CaseをＸＭＬ形式で記述した例を示す説明図である。なお、図５２～図５５には、図５０、図５１に示したD-Caseから一部を抜粋して示している。図５６は、図５０、図５１、および図５２～図５５に示したD-Caseから変換した監視制御データ（障害対応スクリプト）の例を示す説明図である。

【０４０５】

図５２～図５５に示すように、本具体例では、D-CaseのＸＭＬファイルに障害監視制御部３０を制御する障害対応スクリプトが埋めこまれている。そして、図５６が、D-CaseのＸＭＬファイル（図５２～図５５）の「Monitor:M_1」～「Monitor:M_5」からそれぞれ抽出されたスクリプトである。なお、図５０および図５１は、図５２～図５５をグラフィカルに表示したものであり、データには障害対応スクリプトも含まれている。

【０４０６】

このように、D-Caseに障害対応スクリプトが埋め込まれている点が、図４５～図４９を参照しながら説明した、D-Caseパターン<=>モジュール対応テーブル具体例とは異なっている。しかしながら、ステークホルダの合意を示すD-Caseと、監視制御対象システムの障害対応スクリプトとの間を無矛盾に更新することができる点では同じである。

【０４０７】

本発明は以下のように構成してもよい。

【０４０８】

本発明のディペンダビリティ維持装置２０は、監視制御対象システム２の状態を監視し

、必要な場合に対策を実行する障害監視制御部３０の動作を制御する障害監視制御用データ（監視制御データ）を生成し、上記障害監視制御部３０に供給するディペンダビリティ維持装置２０であって、上記監視制御対象システム２のディペンダビリティに関する仕様を記述したディペンダビリティ記述データ（D-Case）をディペンダビリティ記述格納部（D-Case格納部１０）から読み出し、読み出したディペンダビリティ記述データから上記障害監視制御用データを生成するディペンダビリティ記述変換部（D-Case変換部２１）を備えて構成されていてもよい。

【０４０９】

また、本発明のディペンダビリティ維持装置２０の制御方法は、監視制御対象システム２の状態を監視し、必要な場合に対策を実行する障害監視制御部３０の動作を制御する障害監視制御用データを生成し、上記障害監視制御部３０に供給するディペンダビリティ維持装置２０の制御方法であって、上記監視制御対象システム２のディペンダビリティに関する仕様を記述したディペンダビリティ記述データ（D-Case）をディペンダビリティ記述格納部（D-Case格納部１０）から読み出す読み出しステップ（Ｓ２）と、読み出したディペンダビリティ記述データから上記障害監視制御用データを生成する変換ステップ（Ｓ３）とを含む方法であってもよい。

【０４１０】

さらに、上記ディペンダビリティ維持装置２０は、上記障害監視制御部３０は、上記監視制御対象システム２の状態を監視し、必要な場合に対策を実行するために使用するモジュールを、複数のモジュール（モニタモジュール群５０、アクションモジュール群７０）から選択して制御可能であり、上記ディペンダビリティ記述データは、１つの記述（エビデンス）が、上記モジュールを特定するためのモジュール特定情報（変数$2；パターン）を値として設定可能な変数を少なくとも含む形式であり、上記ディペンダビリティ記述変換部は、あらかじめ設定された、上記モジュール特定情報と上記障害監視制御用データとの対応関係を示す情報（D-Caseパターン<=>モジュール対応テーブル）に基づいて、上記ディペンダビリティ記述データに含まれる変換対象の記述を、該記述に含まれるモジュール特定情報に対応する障害監視制御用データに変換するように構成されていてもよい。

【０４１１】

さらに、上記ディペンダビリティ維持装置２０は、上記ディペンダビリティ記述データの１つの記述から変数部分を除いた固定部分に対応して、上記対応関係が設定されており、上記ディペンダビリティ記述変換部は、上記ディペンダビリティ記述データの記述に含まれる上記固定部分に対応する上記対応関係を参照して、該記述を対応する障害監視制御用データに変換するように構成されていてもよい。

【０４１２】

また、本発明の障害対応システム１は、上記のディペンダビリティ維持装置２０と、上記ディペンダビリティ記述格納部（D-Case格納部１０）と、上記障害監視制御部３０とを含み、上記ディペンダビリティ維持装置２０が上記ディペンダビリティ記述格納部から読み出したディペンダビリティ記述データから生成した障害監視制御用データに従って、上記障害監視制御部３０が動作することにより、上記監視制御対象システム２の状態を監視し、必要な場合に対策を実行するように構成されていてもよい。

【０４１３】

さらに、上記障害対応システム１は、上記監視制御対象システム２の状態を監視する１以上のモニタモジュール（モニタモジュール群５０）と、必要な場合に上記監視制御対象システム２に対して対策を実行する１以上のアクションモジュール（アクションモジュール群７０）と、上記障害監視制御部３０の制御により、上記モニタモジュールの選択および動作の制御を行うモニタ管理部４０と、上記障害監視制御部３０の制御により、上記アクションモジュールの選択および動作の制御を行うアクション管理部６０と、を含んで構成されていてもよい。

【０４１４】

また、本発明は以下のように構成してもよい。

【０４１５】
本発明に係るコンピュータシステム（障害対応システム１）は、ステークホルダ間のディペンダビリティに関する合意を記述したディペンダビリティ記述データ（D-Case）を格納するディペンダビリティ記述格納部（D-Case格納部１０）と、対象システム（監視制御対象システム２）内の状態を監視（モニタ）し必要な場合に対策（アクション）を実行する障害監視制御部（障害監視制御部３０）と、障害監視制御部の動作を制御する障害監視制御用データ（監視制御データ）をディペンダビリティ記述データから生成するディペンダビリティ記述変換部（D-Case変換部２１）とを持ち、障害監視制御用データが常にディペンダビリティ記述格納部に格納されたディペンダビリティ記述データからディペンダビリティ記述変換部によって生成されることによりディペンダビリティ記述と障害監視制御部の動作とが無矛盾に維持されるように構成されていてもよい。
【０４１６】
さらに、上記コンピュータシステムは、ディペンダビリティ記述はその一部分がパターン化されており、ディペンダビリティ記述のパターンと障害監視制御用データの対応を表わすテーブル（D-Caseパターン<=>モジュール対応テーブル）を使用して障害監視制御用データをディペンダビリティ記述から生成するように構成されていてもよい。
【０４１７】
さらに、上記コンピュータシステムは、障害監視制御部はモニタ管理部（モニタ管理部４０）とアクション管理部（アクション管理部６０）を持ち、モニタ管理部は１以上のモニタモジュール（モニタモジュール群５０）を管理し、アクション管理部は１以上のアクションモジュール（アクションモジュール群７０）を管理し、モニタ管理部によるモニタモジュールの選択および動作の制御、並びに、アクション管理部によるアクションモジュールの選択および制御と、ディペンダビリティ記述のパターンとの対応を示すテーブルを使用して障害監視制御用データをディペンダビリティ記述から生成するように構成されていてもよい。
【０４１８】
さらに、上記コンピュータシステムの制御方法は、ステークホルダ間のディペンダビリティに関する合意を記述したディペンダビリティ記述データを格納するディペンダビリティ記述格納ステップ（D-Case格納ステップＳ１）と、障害監視制御部が対象システム内の状態を監視し必要な場合に対策を実行する障害監視制御ステップ（障害監視実行ステップＳ３）と、障害監視制御ステップの動作を制御する障害監視制御用データをディペンダビリティ記述から生成するディペンダビリティ記述変換ステップ（D-Case読み出しステップＳ２、D-Case変換ステップＳ３）とを持ち、障害監視制御用データが常にディペンダビリティ記述格納ステップによって格納されたディペンダビリティ記述データからディペンダビリティ記述変換ステップによって生成されることによりディペンダビリティ記述と障害監視制御部の動作とが無矛盾に維持されるように構成されていてもよい。
【０４１９】
また、障害対応システム１の各ブロック、特にディペンダビリティ維持装置２０のD-Case変換部２１は、ハードウェアロジックによって構成してもよいし、次のようにＣＰＵを用いてソフトウェアによって実現してもよい。
【０４２０】
後者の場合、障害対応システム１は、各機能を実現するプログラムの命令を実行するＣＰＵ（central processing unit）、上記プログラムを格納したＲＯＭ（read only memory）、上記プログラムを展開するＲＡＭ（random access memory）、上記プログラムおよび各種データを格納するメモリ等の記憶装置（記録媒体）などを備えている。そして、本発明の目的は、上述した機能を実現するソフトウェアである障害対応システム１あるいはディペンダビリティ維持装置２０の制御プログラムのプログラムコード（実行形式プログラム、中間コードプログラム、ソースプログラム）をコンピュータで読み取り可能に記録した記録媒体を、上記の障害対応システム１あるいはディペンダビリティ維持装置２０に供給し、そのコンピュータ（またはＣＰＵやＭＰＵ）が記録媒体に記録されているプログ

ラムコードを読み出し実行することによっても、達成可能である。

【０４２１】

上記記録媒体としては、例えば、磁気テープやカセットテープ等のテープ系、フロッピー（登録商標）ディスク／ハードディスク等の磁気ディスクやＣＤ－ＲＯＭ／ＭＯ／ＭＤ／ＤＶＤ／ＣＤ－Ｒ等の光ディスクを含むディスク系、ＩＣカード（メモリカードを含む）／光カード等のカード系、あるいはマスクＲＯＭ／ＥＰＲＯＭ／ＥＥＰＲＯＭ／フラッシュＲＯＭ等の半導体メモリ系などを用いることができる。

【０４２２】

また、上記の障害対応システム１あるいはディペンダビリティ維持装置２０を通信ネットワークと接続可能に構成し、上記プログラムコードを通信ネットワークを介して供給してもよい。この通信ネットワークとしては、特に限定されず、例えば、インターネット、イントラネット、エキストラネット、ＬＡＮ、ＩＳＤＮ、ＶＡＮ、ＣＡＴＶ通信網、仮想専用網（virtual private network）、電話回線網、移動体通信網、衛星通信網等が利用可能である。また、通信ネットワークを構成する伝送媒体としては、特に限定されず、例えば、IEEE1394、ＵＳＢ、電力線搬送、ケーブルＴＶ回線、電話線、ＡＤＳＬ回線等の有線でも、ＩｒＤＡやリモコンのような赤外線、Bluetooth（登録商標）、802.11無線、ＨＤＲ、携帯電話網、衛星回線、地上波デジタル網等の無線でも利用可能である。なお、本発明は、上記プログラムコードが電子的な伝送で具現化された、搬送波に埋め込まれたコンピュータデータ信号の形態でも実現され得る。

【０４２３】

上記説明では、機能ブロックおよび手順を図示し参照したが、機能の分離併合または手順の移動は上記機能を満たす限り可能であり、上記説明が本発明を限定するものではない。

【０４２４】

（１）本発明に係るディペンダビリティ維持装置は、監視制御対象システムの状態を監視し、必要な場合に対策を実行する障害監視制御部の動作を制御する障害監視制御用データを生成し、上記障害監視制御部に供給するディペンダビリティ維持装置であって、上記監視制御対象システムのディペンダビリティに関する（ステークホルダ群によって合意された）要求・仕様を記述したディペンダビリティ記述データをディペンダビリティ記述格納部から読み出し、読み出したディペンダビリティ記述データから上記障害監視制御用データを生成するディペンダビリティ記述変換部を備えたことを特徴としている。

【０４２５】

（６）また、本発明に係るディペンダビリティ維持装置の制御方法は、監視制御対象システムの状態を監視し、必要な場合に対策を実行する障害監視制御部の動作を制御する障害監視制御用データを生成し、上記障害監視制御部に供給するディペンダビリティ維持装置の制御方法であって、上記監視制御対象システムのディペンダビリティに関する（ステークホルダ群によって合意された）要求・仕様を記述したディペンダビリティ記述データをディペンダビリティ記述格納部から読み出す読み出しステップと、読み出したディペンダビリティ記述データから上記障害監視制御用データを生成する変換ステップとを含むことを特徴としている。

【０４２６】

上記の構成によれば、ディペンダビリティ維持装置がディペンダビリティ記述格納部から読み出したディペンダビリティ記述データから障害監視制御用データを生成する。そして、これに従って障害監視制御部が動作することにより、障害監視制御部が監視制御対象システムの状態を監視し、必要な場合に対策を実行することができる。

【０４２７】

ここで、ペンダビリティ記述データは、監視制御対象システムのディペンダビリティに関わる仕様を記述したものである。そして、監視制御対象システムの各ステークホルダが監視制御対象システムのディペンダビリティに関して合意した際、その結果がペンダビリティ記述データとして記述され、ディペンダビリティ記述格納部に格納されたものである

ことが望ましい。

【０４２８】

このように、障害監視制御部が用いる障害監視制御用データが、ディペンダビリティ記述格納部に格納されたディペンダビリティ記述データから、ディペンダビリティ維持装置によって生成されることにより、ディペンダビリティ記述データと障害監視制御部の動作とが常に無矛盾になるように維持することが可能となる。

【０４２９】

（２）さらに、本発明に係るディペンダビリティ維持装置は、上記（１）に記載の構成において、上記障害監視制御部は、上記監視制御対象システムの状態を監視し、必要な場合に対策を実行するために使用するモジュールを、複数のモジュールから選択して制御可能であり、上記ディペンダビリティ記述データは、１つの記述が、上記モジュールを特定するためのモジュール特定情報を値として設定可能な変数を少なくとも含む形式であり、上記ディペンダビリティ記述変換部は、あらかじめ設定された、上記モジュール特定情報と上記障害監視制御用データとの対応関係を示す情報に基づいて、上記ディペンダビリティ記述データに含まれる変換対象の記述を、該記述に含まれるモジュール特定情報に対応する障害監視制御用データに変換することを特徴としている。

【０４３０】

上記の構成によれば、さらに、ディペンダビリティ記述データに含まれる変換対象の記述を障害監視制御用データに変換する際、該記述に含まれるモジュール特定情報によって特定されるモジュールを障害監視制御部に選択させ、制御させる障害監視制御用データを生成する。

【０４３１】

これにより、障害監視制御部が制御可能なモジュールが複数ある場合であっても、ディペンダビリティ記述データの記述から適切なモジュールを制御対象とした障害監視制御用データを生成することが可能となる。

【０４３２】

（３）さらに、本発明に係るディペンダビリティ維持装置は、上記（２）に記載の構成において、上記ディペンダビリティ記述データの１つの記述から変数部分を除いた固定部分に対応して、上記対応関係が設定されており、上記ディペンダビリティ記述変換部は、上記ディペンダビリティ記述データの記述に含まれる上記固定部分に対応する上記対応関係を参照して、該記述を対応する障害監視制御用データに変換することを特徴としている。

【０４３３】

上記の構成によれば、さらに、ディペンダビリティ記述データに含まれる変換対象の記述を障害監視制御用データに変換する際、該記述からに含まれる変数部分を除いた固定部分に対応した対応関係を参照して、該記述を対応する障害監視制御用データに変換する。

【０４３４】

これにより、ディペンダビリティ記述データに含まれる記述の固定部分毎に、異なる対応関係を対応付けておくことが可能となる。よって、モジュール特定情報等の変数が同じであっても、固定部分が異なれば、異なる対応関係を用いて、ディペンダビリティ記述データに含まれる記述を障害監視制御用データに変換することができる。それゆえ、多様な変換を簡易な規則によって実現することができる。

【０４３５】

（４）また、本発明に係る障害対応システムは、上記（１）から（３）のいずれかに記載のディペンダビリティ維持装置と、上記ディペンダビリティ記述格納部と、上記障害監視制御部とを含み、上記ディペンダビリティ維持装置が上記ディペンダビリティ記述格納部から読み出したディペンダビリティ記述データから生成した障害監視制御用データに従って、上記障害監視制御部が動作することにより、上記監視制御対象システムの状態を監視し、必要な場合に対策を実行するように構成してもよい。

【０４３６】

（５）さらに、本発明に係る障害対応システムは、上記（４）に記載の構成において、上記監視制御対象システムの状態を監視する１以上のモニタモジュールと、必要な場合に上記監視制御対象システムに対して対策を実行する１以上のアクションモジュールと、上記障害監視制御部の制御により、上記モニタモジュールの選択および動作の制御を行うモニタ管理部と、上記障害監視制御部の制御により、上記アクションモジュールの選択および動作の制御を行うアクション管理部と、を含むように構成してもよい。

【０４３７】

なお、上記のディペンダビリティ維持装置および障害対応システムは、コンピュータによって実現してもよく、この場合には、コンピュータを上記ディペンダビリティ記述変換部として動作させることにより上記ディペンダビリティ維持装置をコンピュータにて実現させるディペンダビリティ維持装置のプログラム、およびそれを記録したコンピュータ読み取り可能な記録媒体も、本発明の範疇に入る。

【０４３８】

上記説明では、機能ブロックおよび手順を図示し参照したが、機能の分離併合または手順の移動は上記機能を満たす限り可能であり、上記説明が本発明を限定するものではない。

【０４３９】

発明の詳細な説明の項においてなされた具体的な実施態様または実施例は、あくまでも、本発明の技術内容を明らかにするものであって、そのような具体例にのみ限定して狭義に解釈されるべきものではなく、本発明の精神と次に記載する特許請求事項との範囲内で、いろいろと変更して実施することができるものである。

【産業上の利用可能性】

【０４４０】

本発明のディペンダビリティ維持装置によれば、不完全性と不確実性が潜在的に存在するオープン環境において、ステークホルダ間における要求の誤解、環境の変化に対する対応不能、障害対応の失敗という３つの問題に対するソリューションを提供することができる。オープン環境でのシステム開発及びシステム運用に広く利用することができる。

【０４４１】

また、本発明の障害対応システムおよびディペンダビリティ維持装置によれば、各ステークホルダの合意の記述を容易に行うとともに、監視制御対象システムの一部に更新があった場合に、合意の記述とモジュールの実装とが無矛盾に開発・更新されるように維持することができる。よって、例えば、組込みシステムから、インターネットなどでつながれた多数システムまで、多様なシステムのディペンダビリティを維持するため装置・方法として好適である。

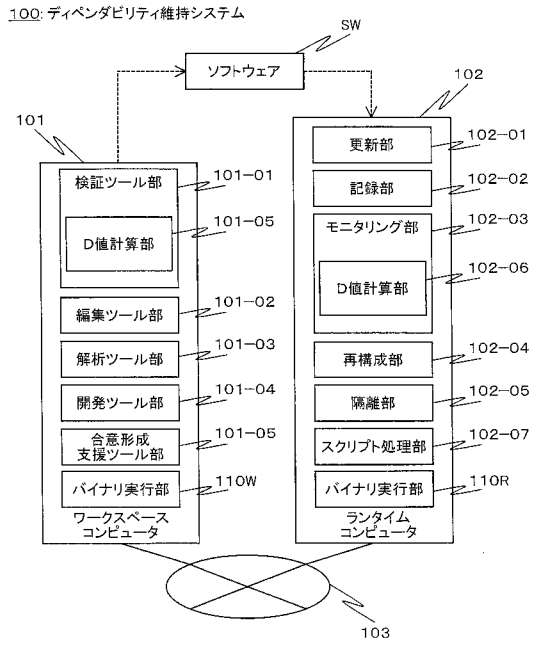
【符号の説明】

【０４４２】

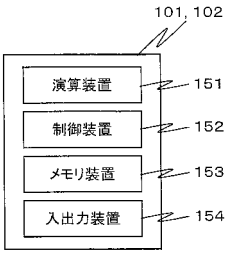
１００（１００Ｕ，１００Ｓ，１００Ｂ，１００Ｐ）　ディペンダビリティ維持システム
１０１　ワークスペースコンピュータ（ディペンダビリティ維持装置、ワークスペース装置、変化対応サイクル実行装置）
１０１－０５　Ｄ値計算部（ディペンダビリティ値決定手段）
１０２　ランタイムコンピュータ（ディペンダビリティ維持装置、ランタイム装置、障害対応サイクル実行装置）
１０２－０４　再構成部（再構成手段）
１０２－０６　Ｄ値計算部（ディペンダビリティ値決定手段）
１０２－０７　スクリプト処理部（スクリプト処理手段）
９０１－０１　ディペンダビリティ記述データ入力部（記述データ取得手段）
９０２－０１　ソフトウェア入力部（記述データ取得手段）
９０２－０２　変更送出部（変更要求送信手段）
Ｐ３００　変化対応サイクル

Ｐ３０１　障害対応サイクル
１　障害対応システム
２　監視制御対象システム
１０　Ｄ－Ｃａｓｅ格納部（ディペンダビリティ記述格納部）
２０　ディペンダビリティ維持装置
２１　Ｄ－Ｃａｓｅ変換部（ディペンダビリティ記述変換部、ディペンダビリティ記述変換手段）
２２　Ｄ－Ｃａｓｅパターン＜＝＞モジュール対応テーブル
３０　障害監視制御部
４０　モニタ管理部
５０　モニタモジュール群（モジュール）
６０　アクション管理部
７０　アクションモジュール群（モジュール）
Ｓ２　読み出しステップ
Ｓ３　変換ステップ

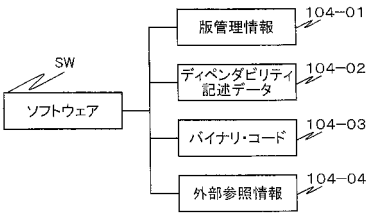
【図１】



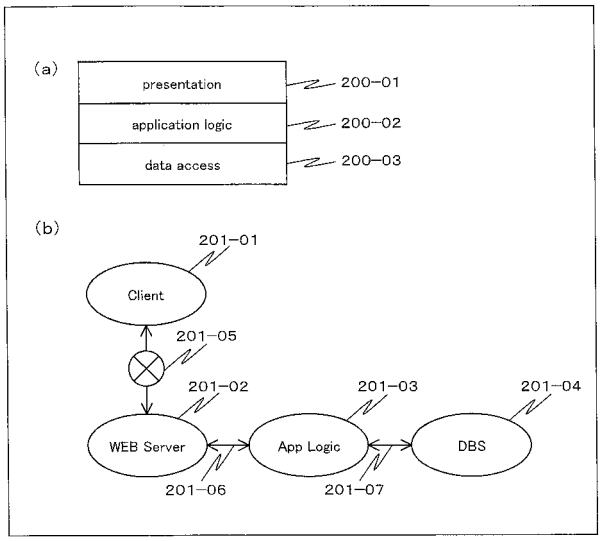
【図２】



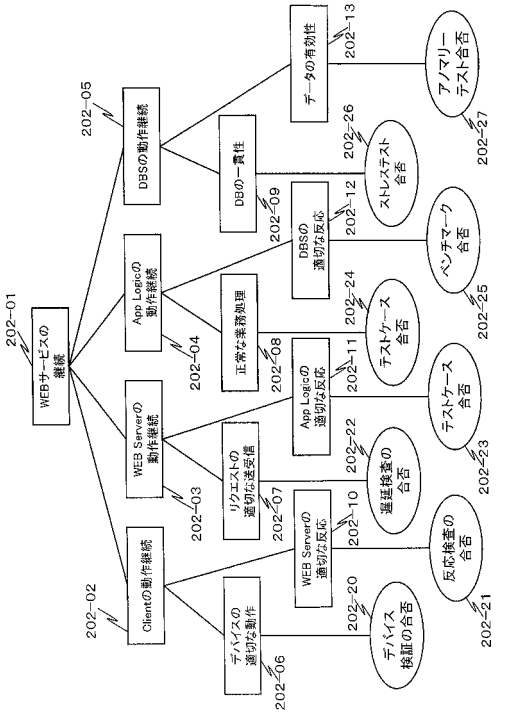
【図３】



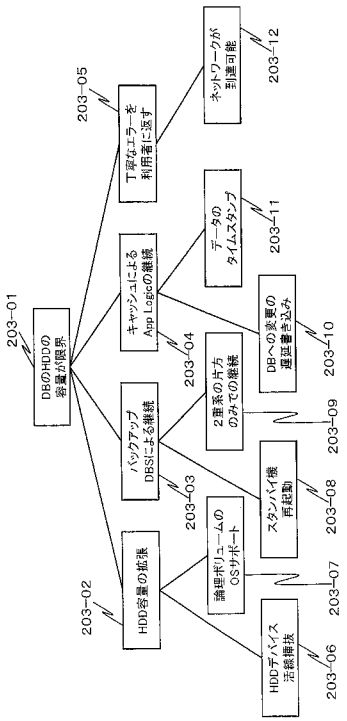
【図４】


(a)
presentation
application logic
data access
200-01
200-02
200-03
(b)
201-01
Client
201-05
201-02
201-03
201-04
WEB Server
App Logic
DBS
201-06
201-07

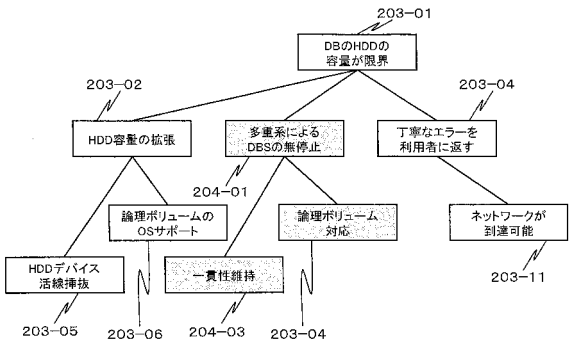
【図５】


202-01
WEBサービスの継続
202-05
DBSの動作継続
202-04
App Logicの動作継続
202-03
WEB Serverの動作継続
202-02
Clientの動作継続
データの有効性
202-13
アノマリーテスト合否
202-27
DBSの一貫性
202-09
202-26
ストレステスト合否
DBSの適切な反応
202-12
ベンチマーク合否
202-25
正常な業務処理
202-08
202-24
テストケース合否
App Logicの適切な反応
202-11
テストケース合否
202-23
リクエストの適切な送受信
202-07
202-22
遅延検査の合否
WEB Serverの適切な反応
202-10
反応検査の合否
202-21
デバイスの適切な動作
202-06
202-20
デバイス検証の合否

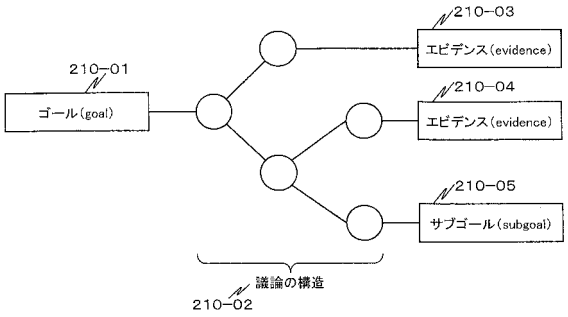
【図６】


203-01
DBのHDDの容量が限界
203-05
丁寧なエラーを利用者に返す
203-12
ネットワークが到達可能
キャッシュによるApp Logicの継続
データのタイムスタンプ
203-11
203-04
2重系の片方のみでの継続
DBへの変更の遅延書き込み
203-10
バックアップDBSによる継続
203-09
203-03
スタンバイ機再起動
203-08
論理ボリュームのOSサポート
203-07
203-02
HDD容量の拡張
HDDデバイス活線挿抜
203-06

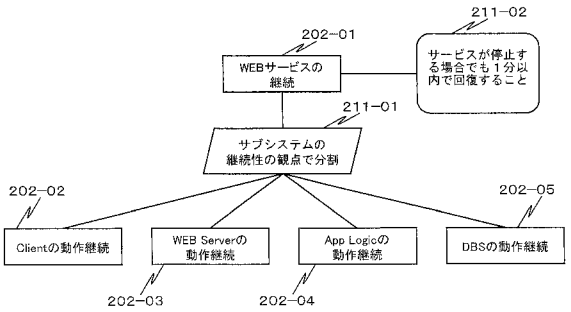
【図７】


203-01
DBのHDDの容量が限界
203-02
HDD容量の拡張
多重系によるDBSの無停止
203-04
丁寧なエラーを利用者に返す
204-01
論理ボリュームのOSサポート
論理ボリューム対応
ネットワークが到達可能
203-11
HDDデバイス活線挿抜
一貫性維持
203-05
203-06
204-03
203-04

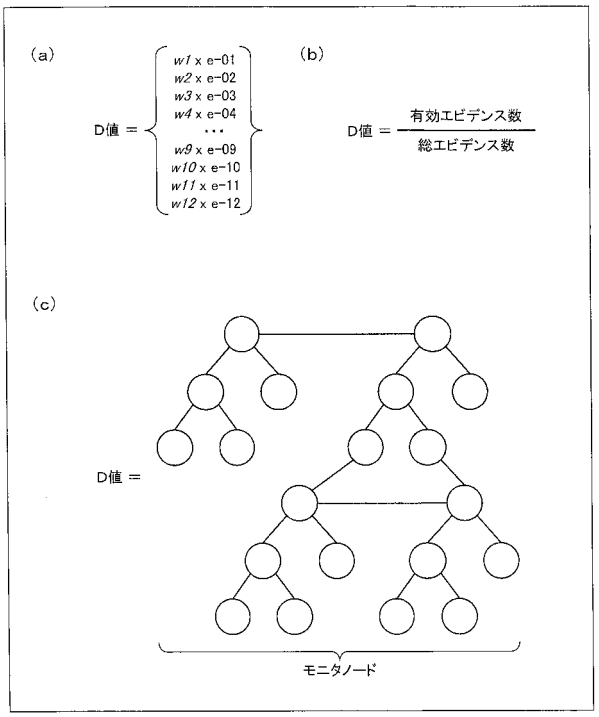
【図８】


210-03
エビデンス(evidence)
210-01
ゴール(goal)
210-04
エビデンス(evidence)
210-05
サブゴール(subgoal)
議論の構造
210-02

【図９】



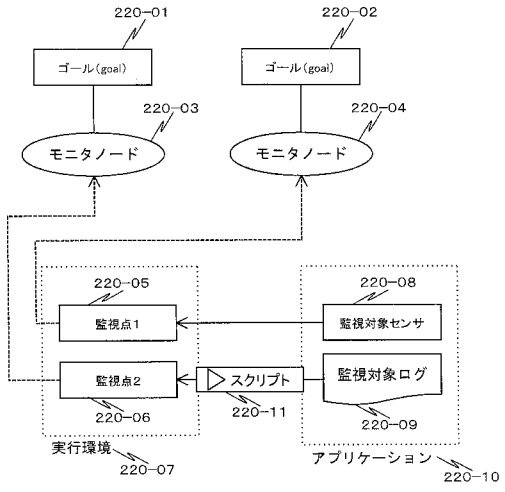
【図１０】



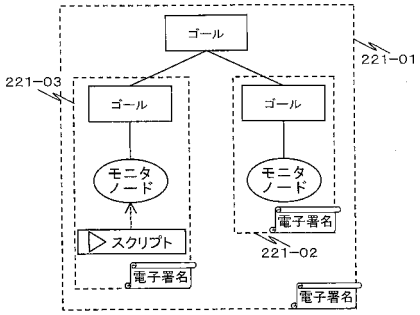
【図１１】

異常の種類	目的	フェーズ	ターゲット
自然災害	可用性	仕様	CPU
人的エラー	信頼性	設計	メモリ
悪意あるアタック	安全性	統合	ファイルシステム
ハードウェア障害	完全性	検証	ネットワーク
	保守性	導入	I/O
		保守	消費電力

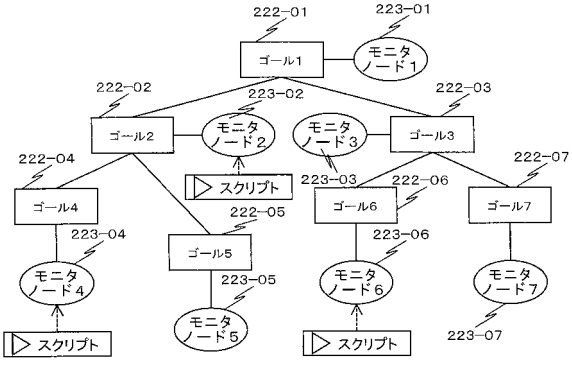
【図１２】



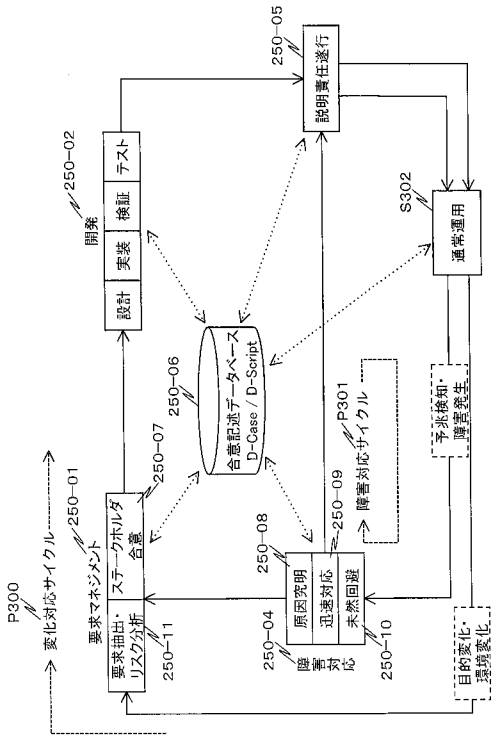
【図１３】



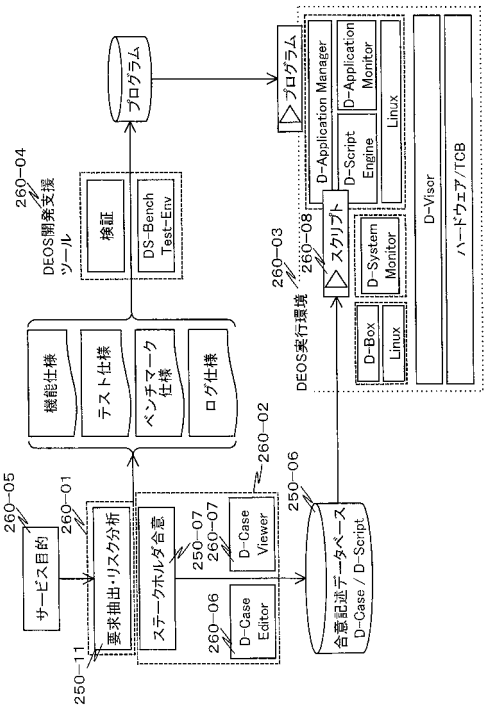
【図１４】



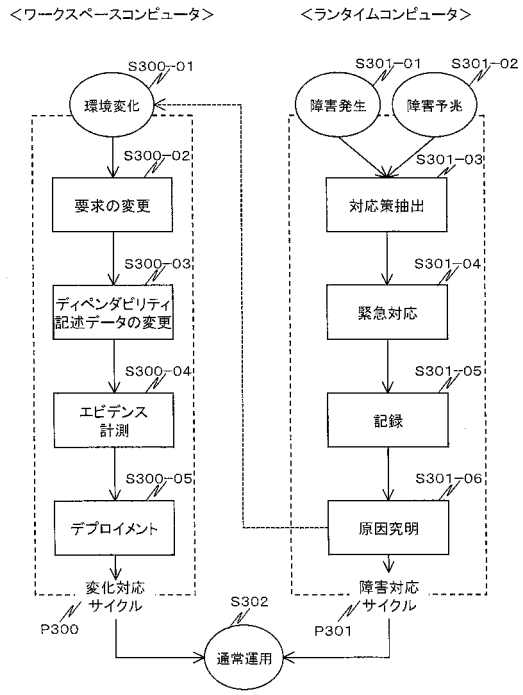
【図１５】



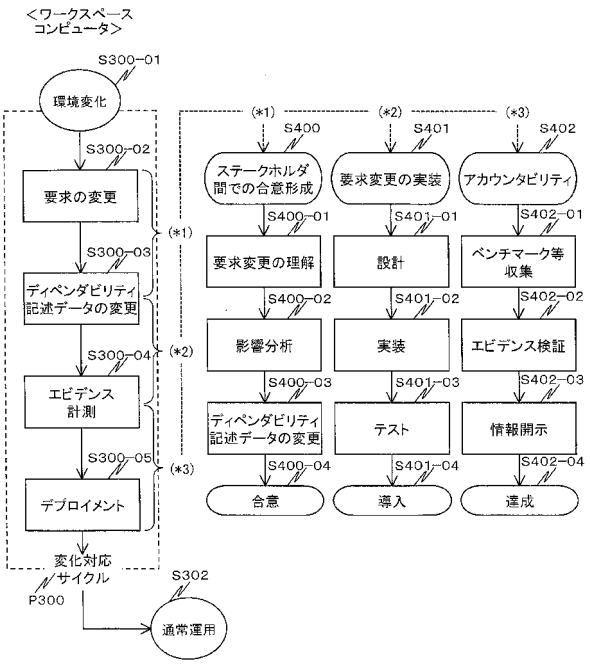
【図１６】



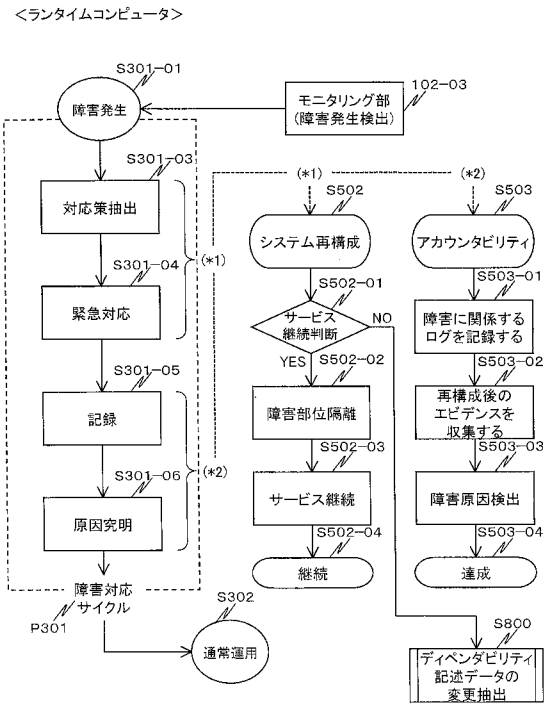
【図１７】



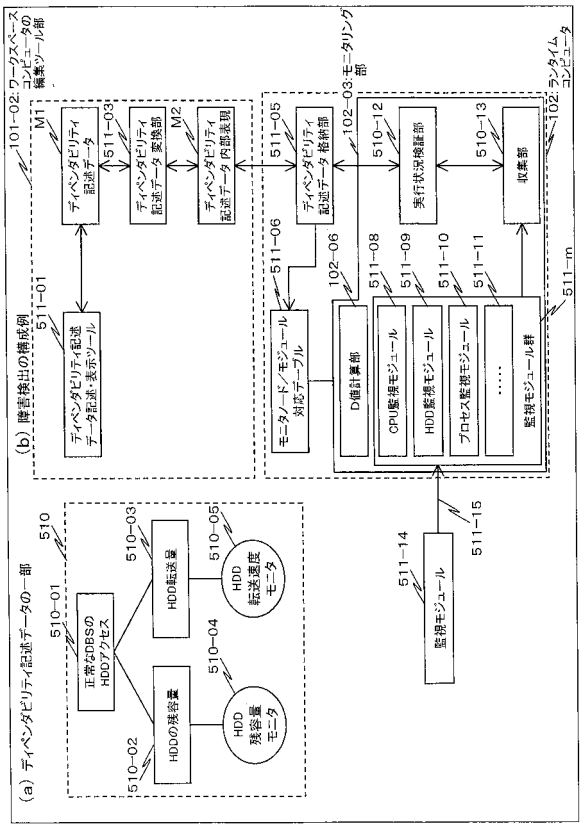
【図１８】



【図１９】



【図２０】

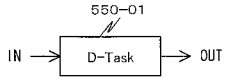


【図２１】

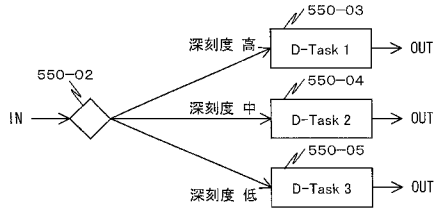
モニタノード	モジュール	
$2	モジュール名称	引数
ディスク容量	HDD監視	$1, $3
ディスク転送量	HDD監視	$1, $3

【図２２】

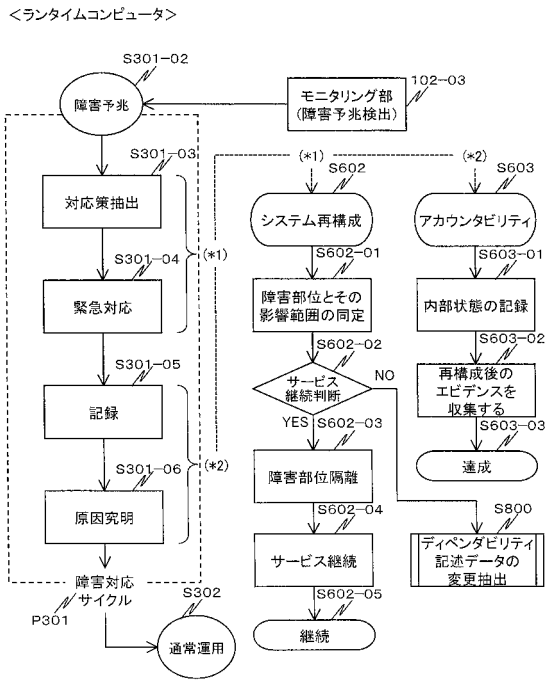
基本動作 (D-Task)



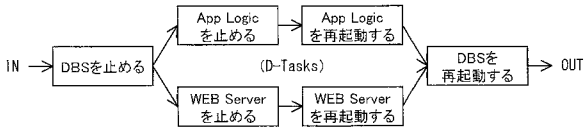
制御動作 (D-Control)



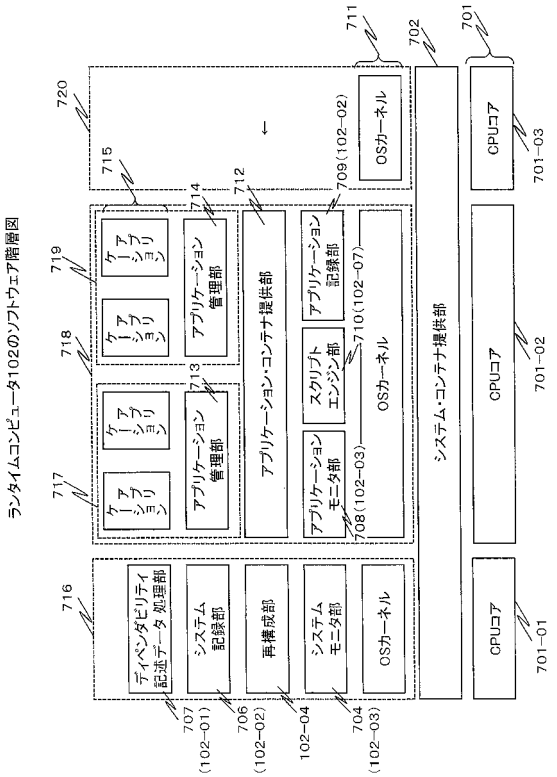
【図２３】



【図２４】



【図２５】

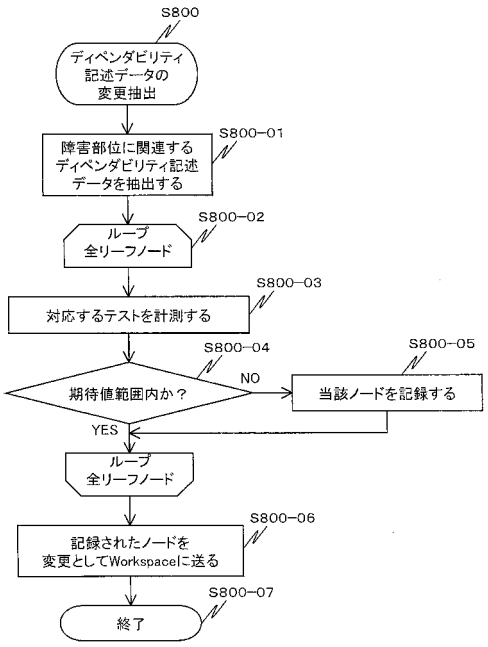


【図２６】

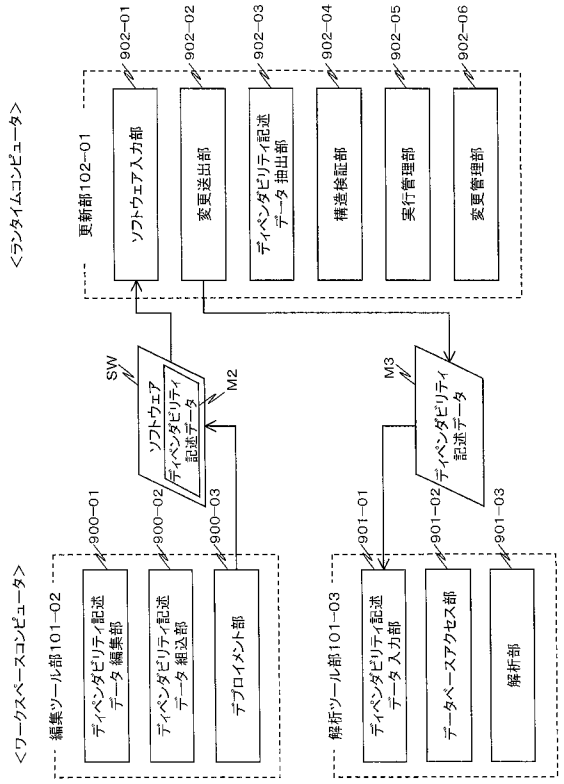
隔離項目に関する機能要件

隔離項目	機能要件
アドレス空間	独立したアドレッシングが可能であること。
名前空間	ファイル名、プロセスID、等のIDが独立していること。
実メモリ	独立した実メモリアクセスが可能であること。
キャッシュメモリ	キャッシュメモリの影響が独立していること。
CPUスケジューリング	CPUを独立して利用できること。
CPUアサインメント	マルチコア構成の場合にCPU割り当てが独立していること。
I/Oバンド幅	I/Oのバンド幅が独立していること。
バス・バンド幅	バスのバンド幅が独立していること。
割り込み	割り込み処理が独立していること。
Time-of-day	日付と時刻が独立していること。
特権	独立した特権が与えられていること。

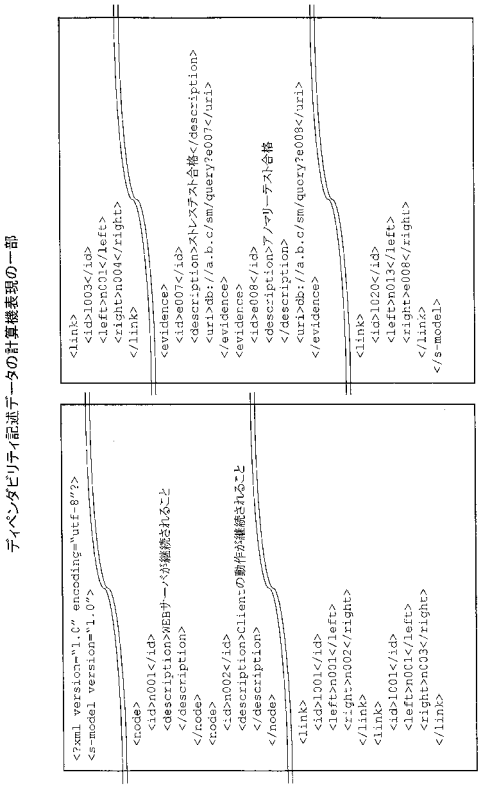
【図２７】



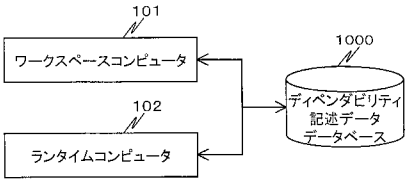
【図２８】



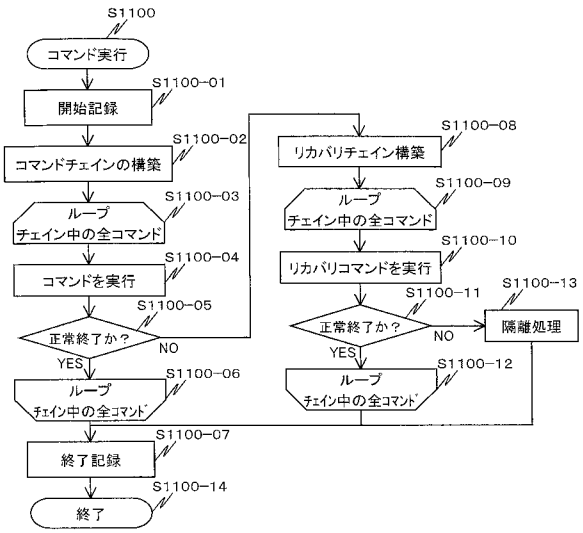
【図２９】



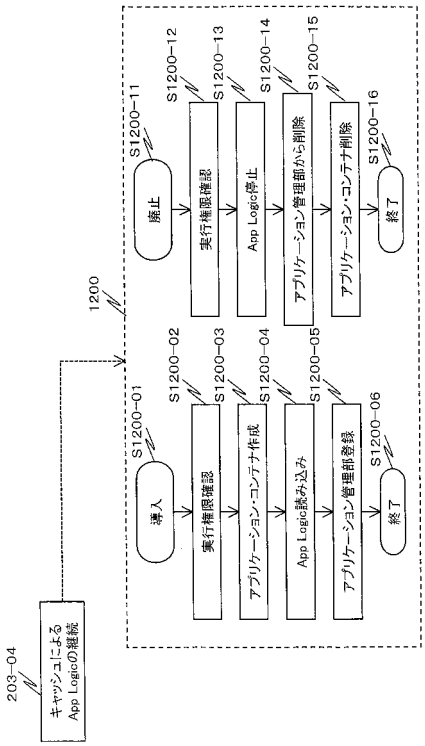
【図３０】



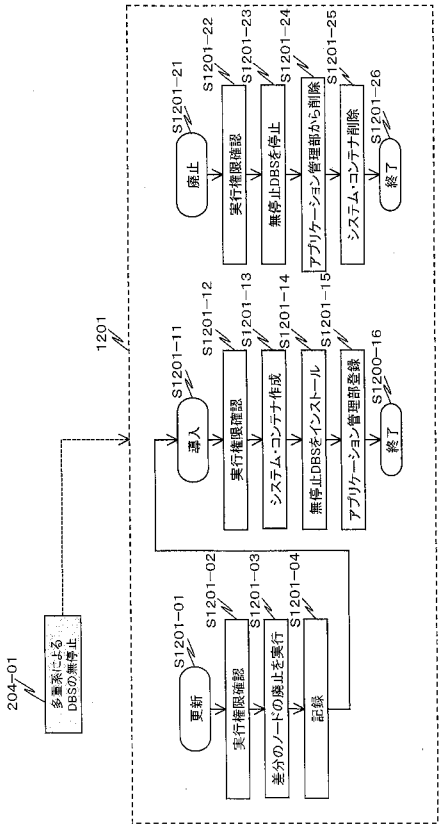
【図３１】



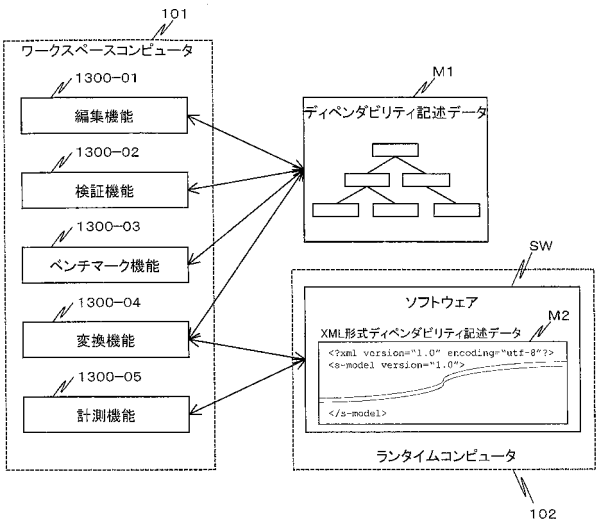
【図３２】



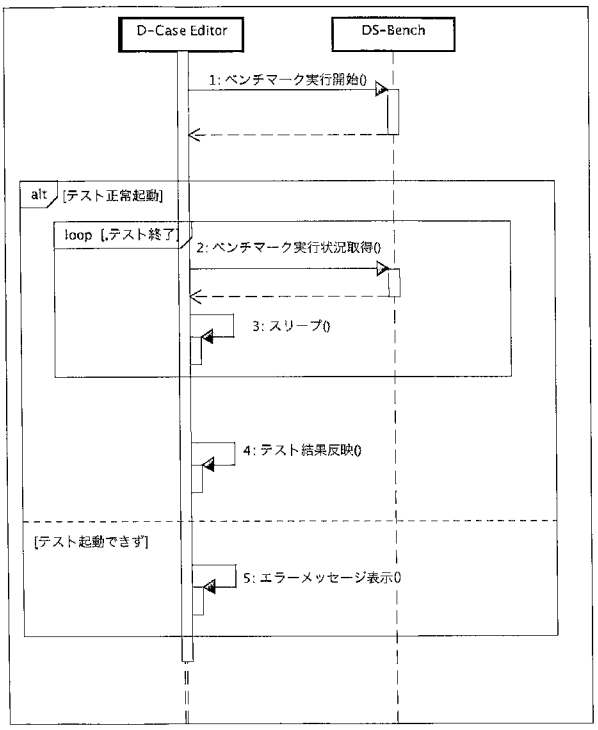
【図３３】



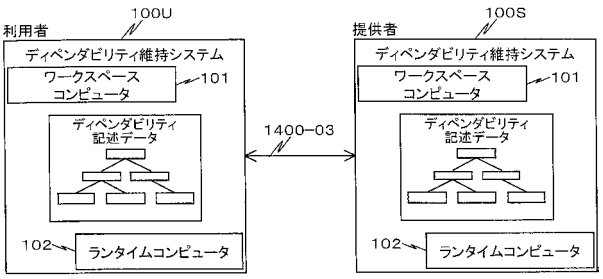
【図３４】



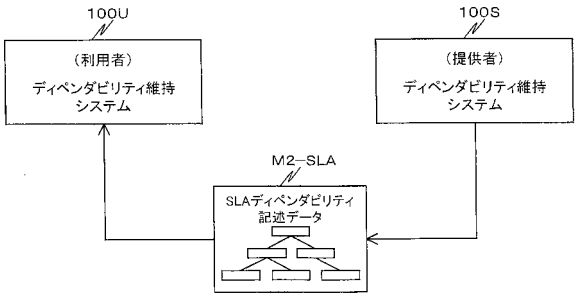
【図３５】



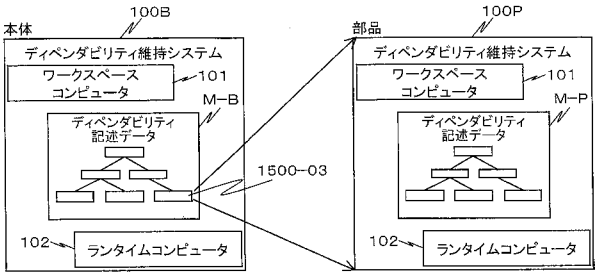
【図３６】



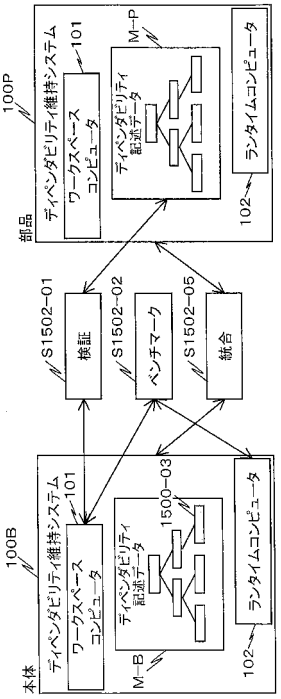
【図３７】



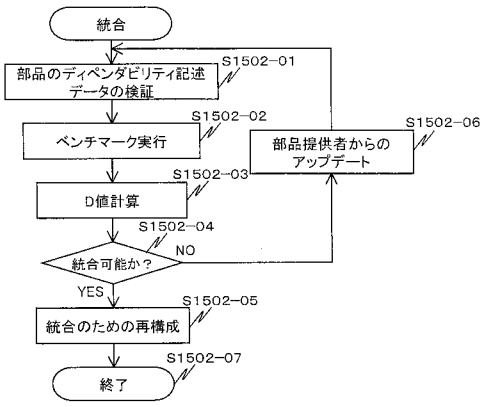
【図３８】



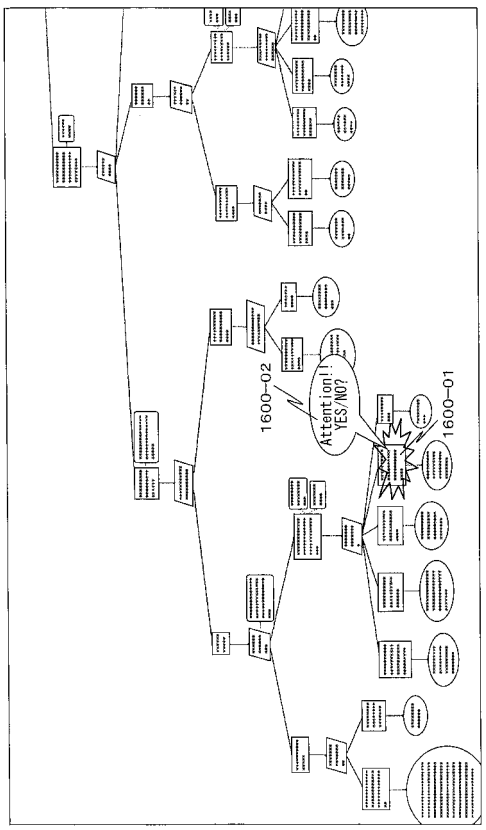
【図３９】



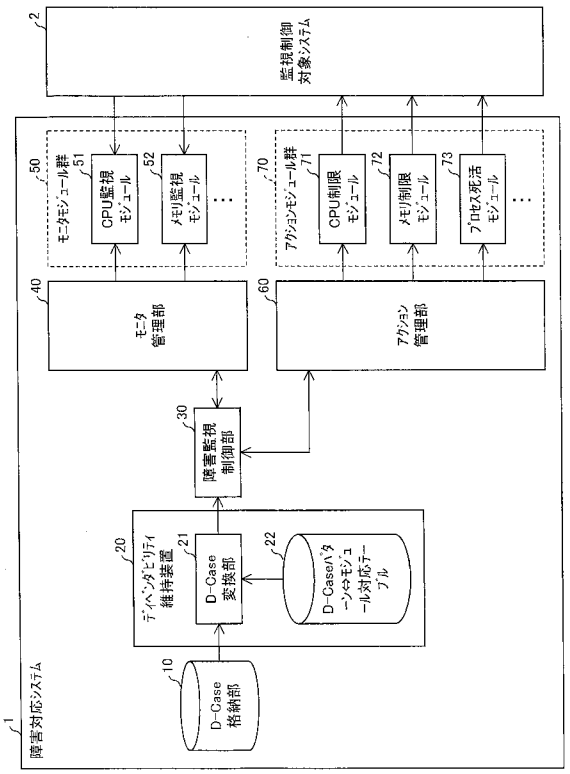
【図４０】



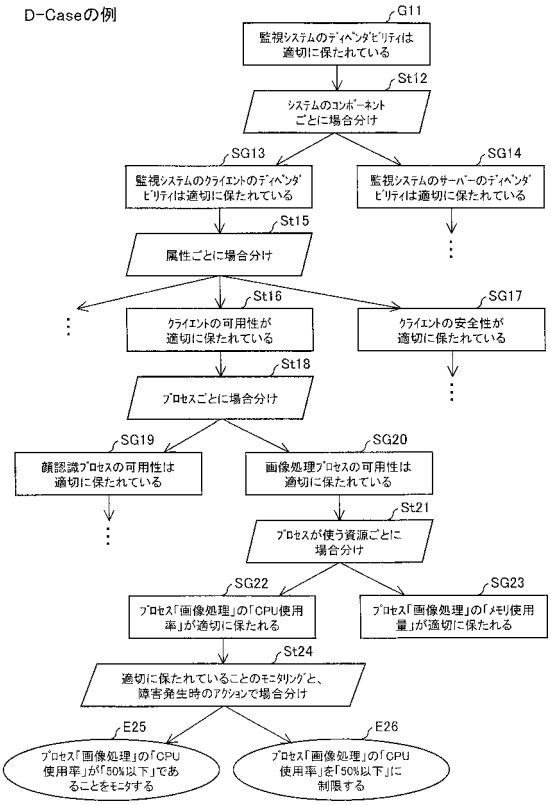
【図４１】



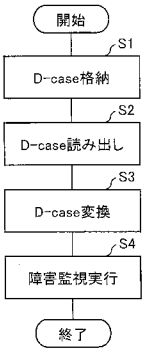
【図４２】



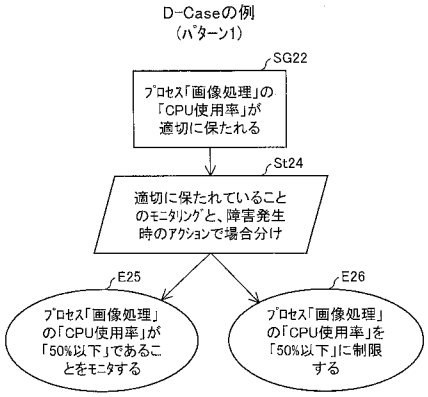
【図４３】



【図４４】



【図４５】



【図４６】

（ａ）

パターン1(モニタモジュール):プロセス「$1」の「$2」が「$3」であることをモニタする

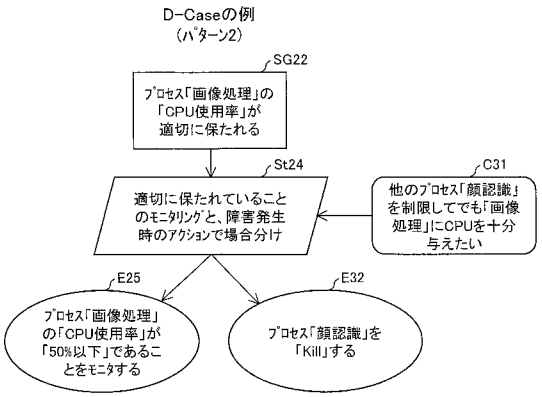
パターン	モジュール	
$2	モジュール名	引数
CPU使用率	CPU監視	$1, $3
メモリ使用量	メモリ監視	$1, $3
ディスク転送量	ディスク監視	$1, $3

（ｂ）

パターン1(アクションモジュール):プロセス「$1」の「$2」を「$3」に制限する

パターン	モジュール	
$2	モジュール名	引数
CPU使用率	CPU制限	$1, $3
メモリ使用量	メモリ制限	$1, $3
ディスク転送量	ディスク制限	$1, $3

【図４７】



【図４８】

（ａ）

ﾊﾟﾀｰﾝ2(ﾓﾆﾀﾓｼﾞｭｰﾙ):ﾌﾟﾛｾｽ「$1」の「$2」が「$3」であることをﾓﾆﾀする

ﾊﾟﾀｰﾝ	ﾓｼﾞｭｰﾙ	
$2	ﾓｼﾞｭｰﾙ名	引数
CPU使用率	CPU監視	$1, $3
ﾒﾓﾘ使用量	ﾒﾓﾘ監視	$1, $3
ﾃﾞｨｽｸ転送量	ﾃﾞｨｽｸ監視	$1, $3

（ｂ）

ﾊﾟﾀｰﾝ2(ｱｸｼｮﾝﾓｼﾞｭｰﾙ):ﾌﾟﾛｾｽ「$1」を「$2」する

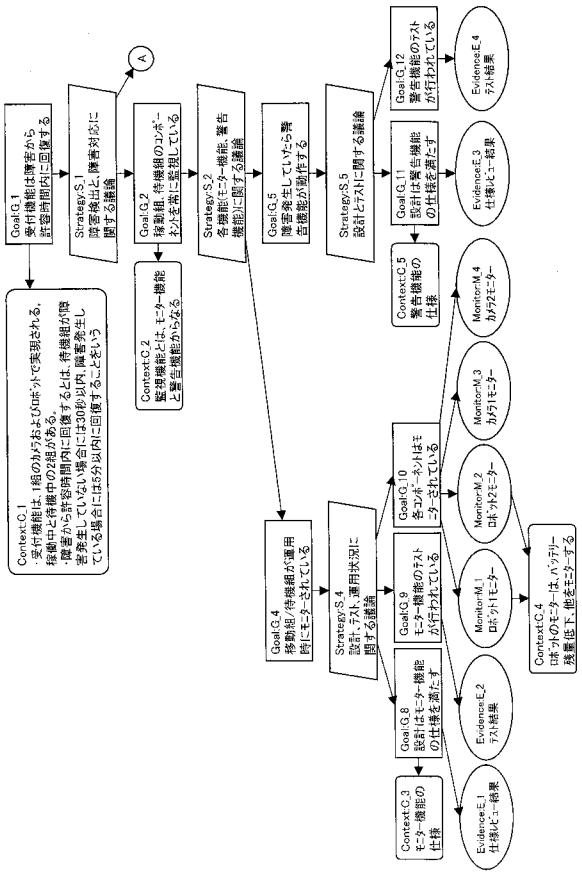
ﾊﾟﾀｰﾝ	ﾓｼﾞｭｰﾙ	
$2	ﾓｼﾞｭｰﾙ名	引数
Kill	ﾌﾟﾛｾｽ死活	Kill, $1
Restart	ﾌﾟﾛｾｽ死活	Restart, $1
Migration	ﾏｲｸﾞﾚｰｼｮﾝ	$1

【図４９】

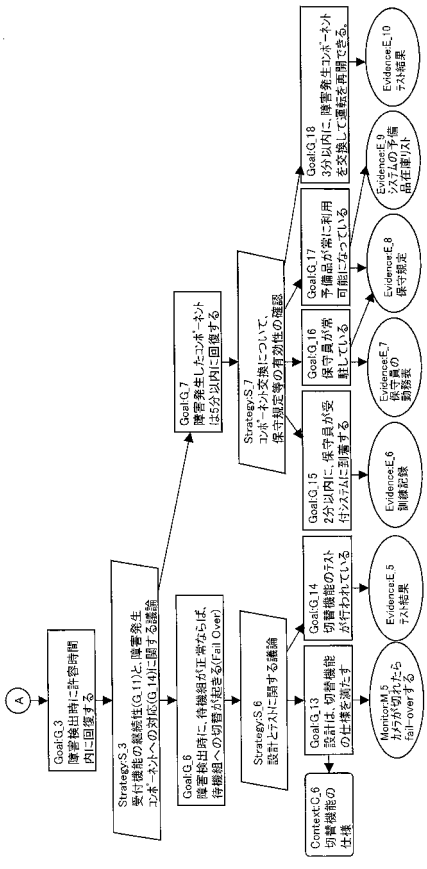
ﾊﾟﾀｰﾝ3(ﾓﾆﾀﾓｼﾞｭｰﾙ):ﾌﾟﾛｾｽ「$1」の「$2」が「$3」であることをﾓﾆﾀする

	ﾊﾟﾀｰﾝ	ﾓｼﾞｭｰﾙ	
	$2	ﾓｼﾞｭｰﾙ名	引数
	CPU使用率	CPU監視	$1, $3
	ﾒﾓﾘ使用量	ﾒﾓﾘ監視	$1, $3
	ﾃﾞｨｽｸ転送量	ﾃﾞｨｽｸ監視	$1, $3
追加→	ﾈｯﾄﾜｰｸ転送量	ﾈｯﾄﾜｰｸ監視	$1, $3

【図５０】



【図５１】



【図５２】

```
<?xml version="1.0" encoding="UTF-8" ?>
- <dcase:dcase xmlns:dcase="http://www.sample_dcase.co.jp/2010/06/dcase"
id="_QdCjcPOIEd-p_d8e3tzhBw">
  <dcase:description />
- <dcase:nodes>
- <dcase:node type="Context" id="id1" name="C_1">
  <dcase:description>・受付機能は、1組のカメラおよびロボットで実現される。 稼働中と待機中の2組がある。・障害から許容時間内に回復するとは、待機組が障害発生していない場合には30秒以内、障害発生している場合には5分以内に回復することをいう。
  </dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Goal" id="id2" name="G_1">
  <dcase:description>受付機能は障害から許容時間内に回復する</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Strategy" id="id3" name="S_1">
  <dcase:description>障害検出と、障害対応に関する議論</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Goal" id="id4" name="G_4">
  <dcase:description>移動組／待機組が運用時にモニターされている</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Strategy" id="id5" name="S_4">
  <dcase:description>設計、テスト、運用状況に関する議論</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Context" id="id6" name="C_3">
  <dcase:description>モニター機能の仕様</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Goal" id="id7" name="G_8">
  <dcase:description>設計はモニター機能の仕様を満たす</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Evidence" id="id8" name="E_1">
  <dcase:description>仕様レビュー結果</dcase:description>
  <dcase:properties />
  </dcase:node>
```

【図５３】

```
- <dcase:node type="Goal" id="id9" name="G_9">
  <dcase:description>モニター機能のテストが行われている</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Evidence" id="id10" name="E_2">
  <dcase:description>テスト結果</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Goal" id="id11" name="G_10">
  <dcase:description>各コンポーネントはモニターされている</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Monitor" id="id12" name="M_1">
  <dcase:description>ロボット1モニター</dcase:description>
- <dcase:properties>
  <dcase:property name="Userdef002" value="main" />
  <dcase:property name="Userdef005" value="ロボット{NUMBER}モニター" />
  <dcase:property name="Userdef006" value="if exec
/home/et2010/damd/RobotCheck.sh main id="{id}" {n}then exec
/home/et2010/damd/RobotError.sh main" />
  <dcase:property name="Userdef007" value="NUMBER=1" />
  </dcase:properties>
  </dcase:node>
- <dcase:node type="Monitor" id="id13" name="M_2">
  <dcase:description>ロボット2モニター</dcase:description>
- <dcase:properties>
  <dcase:property name="Userdef002" value="backup" />
  <dcase:property name="Userdef005" value="ロボット{NUMBER}モニター" />
  <dcase:property name="Userdef006" value="if exec
/home/et2010/damd/RobotCheck.sh backup id="{id}" {n}then exec
/home/et2010/damd/RobotError.sh backup" />
  <dcase:property name="Userdef007" value="NUMBER=2" />
  </dcase:properties>
  </dcase:node>
- <dcase:node type="Monitor" id="id14" name="M_3">
  <dcase:description>カメラ1モニター</dcase:description>
- <dcase:properties>
  <dcase:property name="Userdef002" value="Camera" />
  <dcase:property name="Userdef005" value="カメラ{NUMBER}モニター" />
  <dcase:property name="Userdef006" value="if exec
/home/et2010/damd/CameraCheck.sh {NUMBER} id="{id}" {n}then exec
/home/et2010/damd/CameraStatus.sh {NUMBER} camera-status" />
  <dcase:property name="Userdef007" value="NUMBER=1" />
  <dcase:property name="Userdef008" value="Score=0,Weight=1" />
  </dcase:properties>
  </dcase:node>
```

【図５４】

```
- <dcase:node type="Monitor" id="id22" name="M_4">
  <dcase:description>カメラ2モニター</dcase:description>
- <dcase:properties>
  <dcase:property name="Userdef002" value="Camera" />
  <dcase:property name="Userdef004" value="" />
  <dcase:property name="Userdef005" value="カメラ{NUMBER}モニター" />
  <dcase:property name="Userdef006" value="if exec
/home/et2010/damd/CameraCheck.sh {NUMBER} id="{id}" {n}then exec
/home/et2010/damd/CameraStatus.sh {NUMBER} camera-status" />
  <dcase:property name="Userdef007" value="NUMBER=2" />
  <dcase:property name="Userdef008" value="Score=0,Weight=1" />
  </dcase:properties>
  </dcase:node>
- <dcase:node type="Goal" id="id15" name="G_13">
  <dcase:description>設計は、切替機能の仕様を満たす</dcase:description>
  <dcase:properties />
  </dcase:node>
- <dcase:node type="Monitor" id="id26" name="M_5">
  <dcase:description>カメラが切れたらfail-overする</dcase:description>
- <dcase:properties>
  <dcase:property name="Userdef002" value="OK" />
  <dcase:property name="Userdef005" value="カメラが切れたら{ACTION}する" />
  <dcase:property name="Userdef006" value="if exec /home/et2010/damd/videocheck.sh
id="{id}" {n}then exec /home/et2010/damd/demo1.sh {ACTION} camera-error" />
  <dcase:property name="Userdef007" value="ACTION=fail-over" />
  <dcase:property name="Userdef008" value="Score=0,Weight=1" />
  </dcase:properties>
  </dcase:node>
- <dcase:node type="Context" id="id16" name="C_6">
  <dcase:description>切替機能の仕様</dcase:description>
  <dcase:properties />
  </dcase:node>
  </dcase:nodes>
- <dcase:links>
- <dcase:link id="id17" name="LINK_72" source="id15" target="id26">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
- <dcase:link id="id18" name="LINK_71" source="id11" target="id14">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
```

【図５５】

```
- <dcase:link id="id19" name="LINK_73" source="id11" target="id12">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
- <dcase:link id="id20" name="LINK_74" source="id11" target="id13">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
- <dcase:link id="id21" name="LINK_70" source="id11" target="id22">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
- <dcase:link id="id23" name="LINK_78" source="id12" target="id24">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
- <dcase:link id="id25" name="LINK_79" source="id13" target="id24">
  <dcase:description />
  <dcase:properties />
  </dcase:link>
  </dcase:links>
  </dcase:dcase>
```

【図５６】

```
/* Monitor:M_1 から生成されたスクリプト 「ロボット1モニター」 */
if exec /home/et2010/damd/RobotCheck.sh main id="id12"
then exec /home/et2010/damd/RobotError.sh main

/* Monitor:M_2 から生成されたスクリプト 「ロボット2モニター」 */
if exec /home/et2010/damd/RobotCheck.sh backup id="id13"
then exec /home/et2010/damd/RobotError.sh backup

/* Monitor:M_3 から生成されたスクリプト 「カメラ1モニター」 */
if exec /home/et2010/damd/CameraCheck.sh 1 id="id14"
then exec /home/et2010/damd/CameraStatus.sh 1 camera-status

/* Monitor:M_4 から生成されたスクリプト 「カメラ2モニター」 */
if exec /home/et2010/damd/CameraCheck.sh 2 id="id22"
then exec /home/et2010/damd/CameraStatus.sh 2 camera-status

/* Monitor:M_5 から生成されたスクリプト 「カメラが切れたらfail-overする」 */
if exec /home/et2010/damd/videocheck.sh id="id26"
then exec /home/et2010/damd/demo1.sh fail-over camera-error
```

フロントページの続き

(72)発明者　松野　裕
日本国千葉県柏市あけぼの４－４－５－２０１

審査官　大塚　俊範

(56)参考文献　特開２００９－１６９６５７（ＪＰ，Ａ）
特開２００８－５３７２０３（ＪＰ，Ａ）
特開２０００－９９３５９（ＪＰ，Ａ）
特開２００７－２４９７３９（ＪＰ，Ａ）
DEOS Core Team, Open Systems Dependability Core -Dependability Metrics-, 日本, 科学技術振興機構, ２００９年　9月　4日, 特に、p.1-18の記載を参照, ＵＲＬ, http://www.crest-os.jst.go.jp/topics/deos2009/metrics.pdf

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ　１１／２８－１１／３６
Ｇ０６Ｆ　１１／１４－１１／２０

(54)【発明の名称】ディペンダビリティ維持システム、変化対応サイクル実行装置、障害対応サイクル実行装置、ディペンダビリティ維持システムの制御方法、制御プログラムおよびそれを記録したコンピュータ読み取り可能な記録媒体