Vol.2010-EVA-33 No.1
2010/11/29

システムモデルベースＳＩ支援環境による性能・可用性評価

伊豆倉さやか†向剣文†榊啓†木村大地†矢野尾一男†

システムインテグレーション（SI）における課題の一つとして、システムの品質問題がある。SI の生産性を向上させるためには、システム開発のなるべく早い段階で設計ミスを検知し、リカバリコストを抑える必要がある。我々は、モデルベース SI を導入し、システム構築前に、顧客要求に基づいたモデルの評価を行うためのモデルベース SI 支援環境「CASSI」の開発に取り組んでいる。
本稿では、我々が定義した、モデル記述用の独自記法と顧客要求の評価モデル、及び CASSI を用いた性能・可用性の評価について紹介する。

Evaluation of Performance and Availability based on Model-based System Integration Environment

S.Izukura†, J.Xiang†, H.Sakaki†, D.Kimura†, K.Yanoo†

One of the key issues on IT system integration (SI) is to reduce the risk of development failures. To improve SI productivity, erroneous design should be detected as early as possible, because earlier detection largely saves recovery cost.
We employ Model-Based System Integration (MBSI), which can evaluate the user's requirements before constructing real systems. An in-house MBSI tool named "CASSI" has been developed and is currently under improvement.
In this paper, we introduce our original modeling language, which is an extension of SysML and annotated by parameters for NFR analysis. An example 3-tier web system is used to demonstrate our method with CASSI.

1. はじめに

SI における大きな課題の一つとして、システムの品質問題がある。システムの開発工程は、主に以下の次の 5 つのフェーズに分けられる。
（1）要求分析（2）システム設計（3）構築（4）テスト・検証（5）運用
システムの品質は、設計者のスキルや経験に大きく依存し、設計段階での見積もりが甘いと、顧客要求に合わない質の悪いシステムが出来てしまう。また、このような設計ミスや要求の反映漏れが開発の後の方のフェーズになるまで検出されないと、手戻り工数が増大してしまう。

このような課題を解決する有力な手段としてモデルベース SI が挙げられる。これにより、個々のシステム設計者のスキルによらない均質な設計を行うことが可能となると共に、開発の早い段階で設計ミスなどを検出することができる。

2. モデルベース SI と非機能要求の評価

2.1 モデルベース SI とは

モデルベース SI（Model-Based System Integration : MBSI）とは、モデルベース開発を SI に適用したものである。通常、システムの仕様書や設計書、またはシステムに対する顧客要求などは、Word や Excel などのドキュメントで記述されることが多く、それらを機械的に検証したり、再利用したりすることは難しい。MBSI では、システムを（準）形式的なモデリング言語で記述し、そのモデルの検証や評価を行う。ここで、このシステムに関する様々な情報を記述したモデルを「システムモデル」と呼ぶ。

このように、システムの設計情報を形式的にモデル化しておくことで、顧客要件を満たしているかを、設計段階で機械的に評価することができるため、設計ミスによる手戻り工数を削減すると共に、要件が保証された高品質なシステムを提供することが可能となる。また、既存のシステムモデルを再利用することで、設計工数を削減すると共に、属人性を排除した均質な設計が可能となる。

我々は、MBSI を実現するための社内向けツール CASSI（Computer Aided System model-based System Integration environment）の開発に取り組んでいる。本稿では主に、CASSI を用いた顧客要件に基づくシステムモデルの評価について説明する。

2.2 非機能要求とその評価

顧客要求は、機能要求と非機能要求（Non-Functional Requirements : NFR）に分類することができる。現状、我々が主なターゲットとしているのは、後者（NFR）の評価・

† NEC サービスプラットフォーム研究所

分析である。

NFR は、以下の三つに分類することができる。

- 性能
- ディペンダビリティ
- セキュリティ

さらに、ディペンダビリティとセキュリティを合わせて、可用性・信頼性・安全性・保全性・保守性・機密性に分けることができる [1]。これらの中で我々は、性能と可用性を、SI 分野における特に重要な性質であると位置付けており、本稿では、主にこれらの評価について説明する。

システムの性能は、システムを構成するハードウェアの容量、及び、実装されるソフトウェアの性質に大きく依存する。設計時には、これらハードウェア及びソフトウェアの挙動を予測・考慮した上で、システム全体のアーキテクチャを見積もる必要がある。そのため、設計者の技量に依存した属人的な設計が行われることになり、性能などの見積もりが甘く、顧客要求を満たさないシステムが構築されてしまう恐れがある。また、システムの可用性は、性能とは異なり、テスト段階で検証されにくい。そのため、ネットワークの冗長化し忘れ等のケアレスミスが、実運用の中で明らかになることが多く、それによって発生するシステムの故障や停止が大きな被害につながる恐れがある。

以上のような設計ミスを早い段階で検出し、開発の手戻りや運用時の事故を防ぐためには、MBSI によって事前にモデルの性能や可用性を評価しておくことが有効であると考えられる。また、システムモデルの中に記述されている設計パラメータの値を変更し、カスタマイズすることで、既存のモデルを再利用することが可能となる。ここで、設計パラメータとは、サーバの台数やメモリ容量、リクエストの到着率、RAID 構成などのシステムの NFR に影響する属性のうち、システムごとに可変な値のことである。これにより、システム設計に要する工数を大幅に短縮すると共に、設計者の技量によらない均質な設計を行うことが可能となる。

2.3 CASSI によるシステムモデルの評価

図 1 に、CASSI を用いたシステムモデルの評価と再利用の流れを示す。

まず、様々なシステムの設計情報を、後述（2.4 節）の独自記法を用いてモデル化し、CASSI のリポジトリに蓄える。その際、評価エンジンを用いてあらかじめ NFR の評価を行っておくことで、モデルの記述ミスを防ぐと共に、整合性を保証することができる。

次に、別のシステムにおいて、リポジトリ内のシステムモデルを再利用する場合には、顧客要求に基づいて適切なモデルを抽出し、そのシステムに対する NFR を満たすかどうかの評価を行う。そして、必要に応じて設計パラメータの値を変更してシステムモデルをカスタマイズし、それを基にシステムを構築する。このように、CASSI を用いることで、システムモデルの評価とカスタマイズを機械的に行うことができるようになり、モデルを容易に再利用することが可能となる。

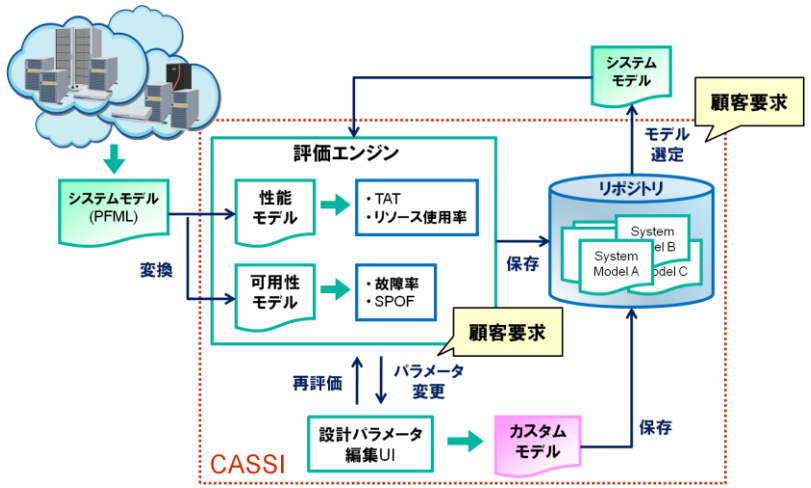

図 1：システムモデルの評価の流れ

2.4 システムモデル記法

我々は、システムモデルを記述するための記法として、SysML[2]を拡張した Plat Form Modeling Language（PFML）を定義した [3]。形式的なモデリング言語としては、SysML 以外に UML や AADL などがよく知られているが、我々は以下の理由から SysML を PFML のベースとして採用した。

- UML をベースとしているため、一般の IT エンジニアにとって AADL よりもなじみが深い
- UML から SysML への拡張の際に導入された Structural Compartment や要求図などの記法が、システムの構造や顧客要求などを記述するのに有用である。

PFML では、システムを構成する機器の接続関係（プラットフォームの設計）を表す構造図と、その上で実行される処理（アプリケーションの動作）を表す振る舞い図、及び各処理がどの機器で実行されているかを表す割当関係を記述する。ここで、プラ

ットフォームには機器だけでなく、OS やミドルウェアとその構成も含める。また、IT 分野に特化した独自プロファイルを定義しており、システムモデルから性能や可用性などの NFR の評価モデルを構築することができる（図 2）。

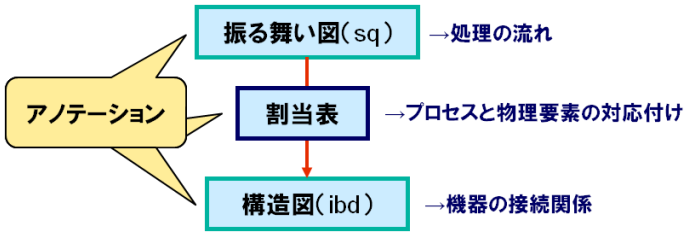

図 2：システムモデル（PFML）の構成

以下に、PFML で記述した構造図（図 3）と振る舞い図（図 4）、及び割当関係（図 5）を記述した例を示す。

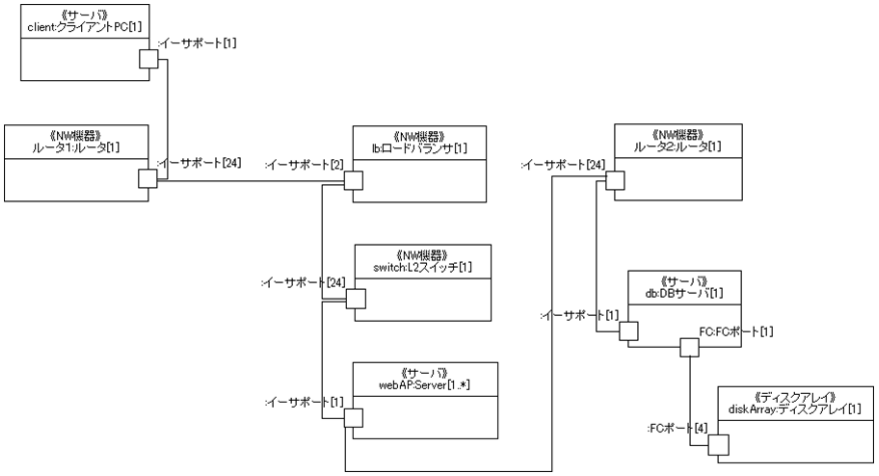

図 3：システムモデル（構造図）の例

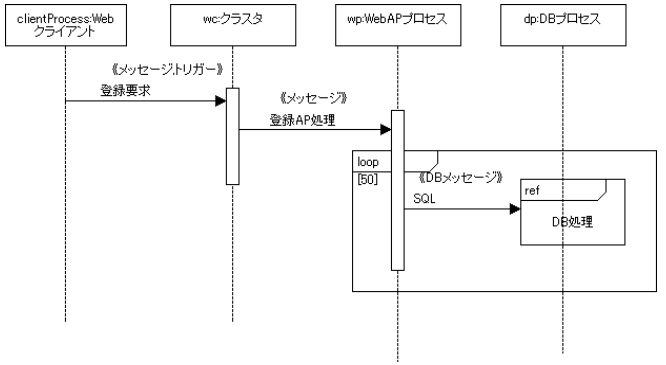

図 4：システムモデル（振る舞い図）の例

構造図は SysML の内部ブロック図、振る舞い図はシーケンス図を用いて記述し、構造図の要素と振る舞い図の要素の間の割当関係を、以下のような表形式で記述する。

実現関係名	関連元	関連先
実現 1	clientProcess:Webクライアント	client:クライアントPC
実現 2	wc:クラスタ	lb:ロードバランサ
実現 3	wp:WebAPプロセス	webAP:Server
実現 4	dp:DBプロセス	db:DBサーバ
実現 5	mv:ボリューム	diskArray:ディスクアレイ
実現 6	iv:ボリューム	diskArray:ディスクアレイ

図 5：システムモデル（割当関係）の例

3. NFR 評価モデル

本章では、PFML で記述したシステムモデルから生成される性能・可用性の評価モデル、及びその変換方法について詳細に述べる。

3.1 性能評価モデル

モデルベースの性能評価手法は、解析的な手法とシミュレーションに分類することができる。本稿では、シミュレーションによる性能評価について説明する。

図 6 は、シミュレーションに用いる性能評価モデルの例である。

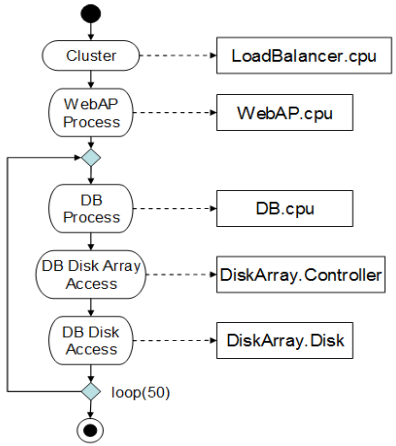

図 6：性能評価モデルの例

本モデルは、システムにおける処理単位（プロセス）を表すノード（アクション）間のフローチャートと、各プロセスがどの機器（リソース）の上で実行されているかを表す割当関係により構成される。このアクション間のフローチャートは、UML のアクティビティ図に相当するものである。

CASSI の評価エンジンは、システムモデルからこの性能評価モデルを自動的に構築し、性能シミュレーションを行っている。

3.1.1 性能評価用プロファイル

PFML では、性能評価モデルを構築するために必要な、IT システムの性能に影響する様々な属性が定義されている。例えば、IT システムならではの属性として、メッセージに対して SQL コマンドの回数やサイズなどを記述している。また、その通信を行う DB サーバに対して SQL 処理性能（スループット等）を記述することができる。

3.1.2 変換アルゴリズム

システムモデルから性能評価モデルへの変換方法は、以下の通りである。

(1) アクティビティモデルの生成

シーケンス図に記述されているプロセスを抽出してアクションを生成し、各アクション間の振る舞いを表すフローチャート（アクティビティモデル）を生成する。

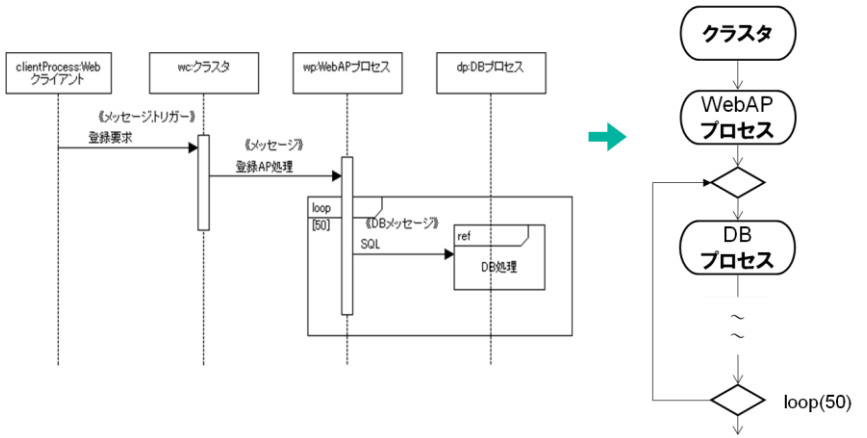

図 7：アクティビティモデルの生成

(2) 割当関係の生成

プロセスとリソースの割当関係から、(1)で生成した各サービスアクションを対応するリソースに割り当てる。

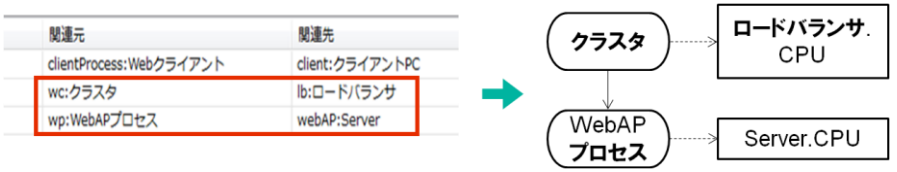

図 8：割当関係の生成

(3) 処理時間の算出

各サービスアクションの処理時間を求める。ここで、処理時間としては、シーケンス図のメッセージに記述されている CPU 時間、または、メッセージサイズをメッセージ受信元のサービスアクションが割り当てられているリソースのスループットで割ったものを用いる。

Vol.2010-EVA-33 No.1
2010/11/29

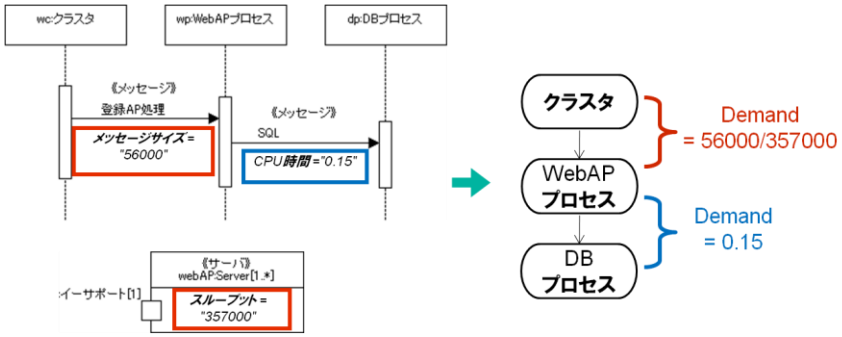

図 9：処理時間の算出

以上のような変換を行うことで、図 6 のようなシミュレーション用の性能評価モデルが構築される。

3.2 可用性評価モデル

本稿で紹介する可用性評価モデルは、システムの復旧プロセスが考慮されていないため、信頼性評価モデルと呼ぶ方がより正確であるが、本稿では可用性評価モデルと総称する。

図 10 は、我々が定義した可用性評価モデル（Reliability Configuration Model : RCM[4]）の例で、二つのクラスタがスター型に結合された構造を表す RCM である。

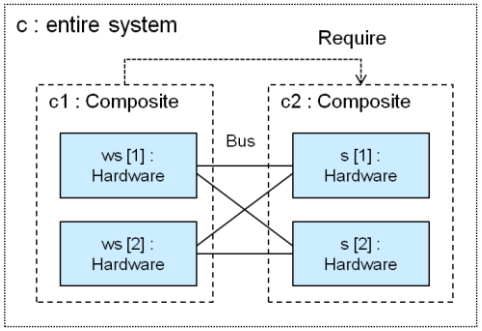

図 10：可用性評価モデル（RCM）の例

この RCM から故障木（FT）を生成することで、可用性の分析を行うことができる。

3.2.1 可用性評価用プロファイル

PFML では、二通りの冗長性の記法が定義されており、これらを用いて可用性評価モデルを構築する。一つは、プロセスとリソース間の割当関係に対して記述された冗長性で、もう一つはリソース間に記述された「スペア」という明示的な冗長性である。前者は、各プロセスがリソース上で実行される際の冗長性を表し、後者は物理モデルに記述されている各リソースが多重化されていることを表すものである。これらの二つの冗長性を考慮して、RCM を構築する。

3.2.2 変換アルゴリズム

システムモデルから可用性評価モデルへの変換方法は、以下の通りである。

(1) Component Tree の生成

各リソースの入れ子関係及び接続関係を表す Component Tree を生成する。ここで、リソースの多重度が 1 以上だったら、Composite とその子供（Hardware）を生成し、リソースが冗長化されている場合には、Composite の代わりに Cluster を生成する。また、コネクタで接続されているリソースから生成された Composite 間に Bus を生成する。

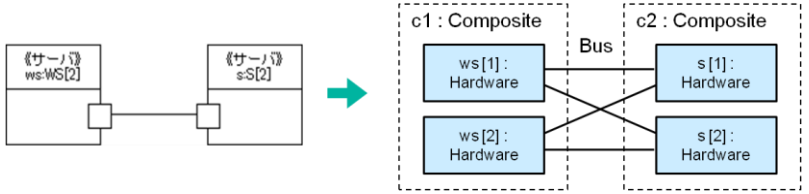

図 11：Component Tree の生成

(2) 構造・機能的依存性の生成

構造図においてスペア関係を持つパーツから生成された Composite 間に hasSpare 関係を生成する。また、シーケンス図でメッセージが送受信されているプロセスから生成された Composite 間に require 関係を生成する。require 関係は、メッセージの送信元のプロセスから受信元のプロセスに向かって生成する。例えば図の require 関係は、メッセージの送信元 wsp が割り当てられているリソース ws から生成された Composite :

c1 から、送信先 sp が割り当てられているリソース s から生成された Composite : c2 へ向かう方向になる。

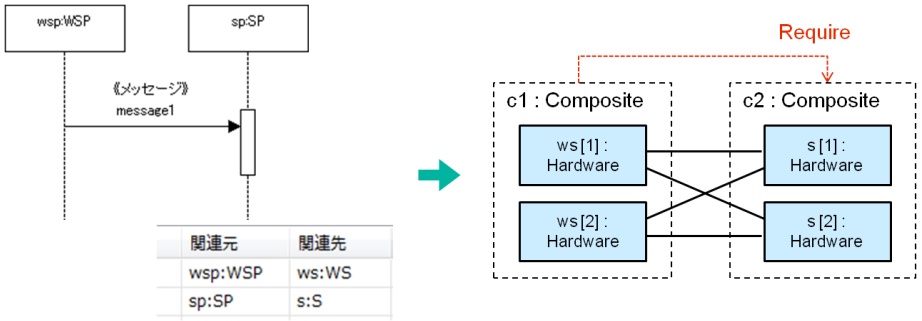

図 12：require 関係の生成

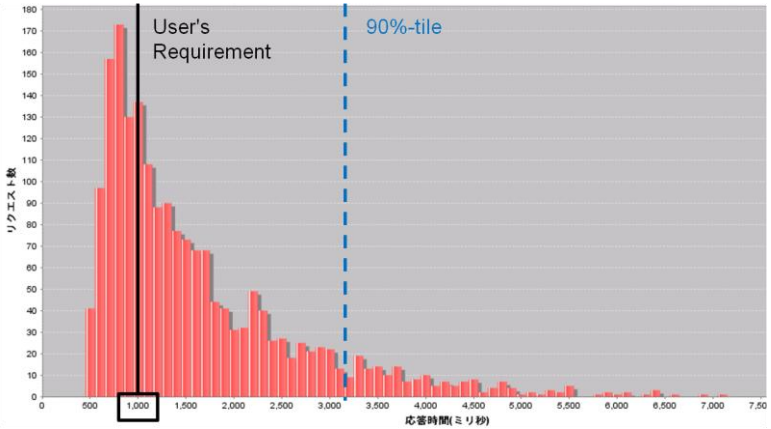

図 13：TAT の分布（WebAP サーバ 2 台）

以上のような変換を行うことで、図 10 のような可用性評価モデルが構築される。このモデルから、故障木（Fault Tree : FT）を生成し、システム全体の故障率の算出や、ミニマルカットセットの分析などを行うことができる [4]。

4. 適用例

本節では、実際に、あるシステムモデルに対して性能シミュレーション及び可用性の評価を行った例を示す。ここでは、評価対象として、第 2.4 節で例として示した一般的な Web3 層システムをモデル化したシステムモデルを取り上げる。ここで、このシステムに対するリクエストの到着率は、ピーク時には約 150(/秒)に達し、顧客要求として、リクエストの応答時間（Turn Around Time：TAT）を 1 秒以内に抑える必要があるとする。

このシステムモデルにおける性能シミュレーションの結果を、以下に示す。図 13 は TAT の分布を示したヒストグラムであり、図 14 はシステムを構成する各リソースの使用率を表す。図 13 より、TAT の 90%Tile の値は約 3 秒で、性能に対する顧客要求を満たしていないことがわかる。この時、図 14 より、WebAP サーバの CPU の使用率が 100%に達しており、ボトルネックとなっていることがわかる。

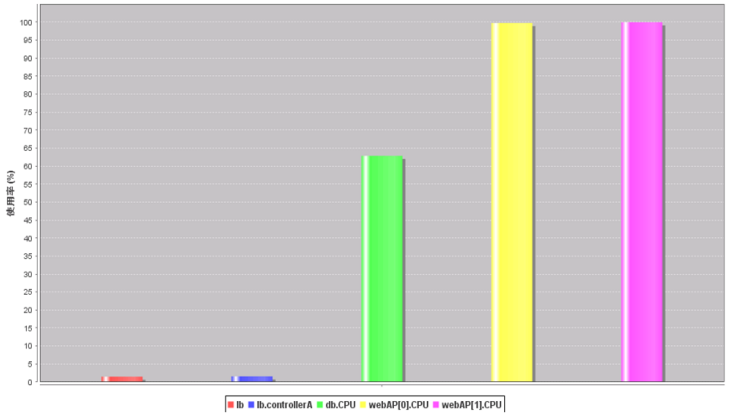

図 14：リソース使用率

ここで、WebAP サーバの台数を 2 台から 4 台に増やした場合のシミュレーション結果は、図 15 のようになる。この場合、TAT の 90%Tile の値は約 0.9 秒となり、顧客要求を満たすことができる。

Vol.2010-EVA-33 No.1
2010/11/29

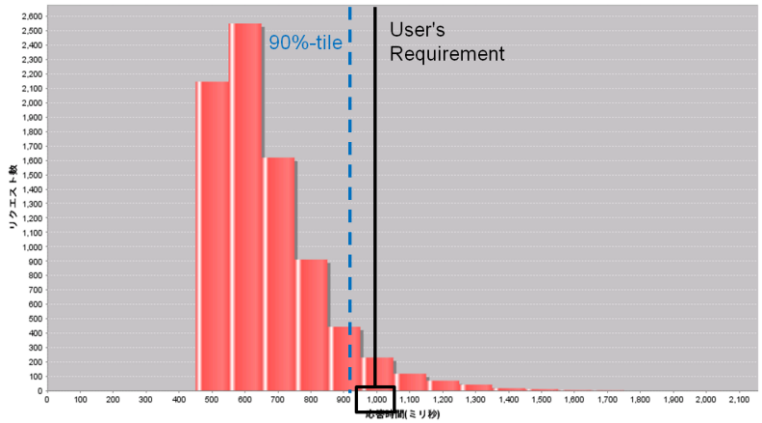

図 15：TAT の分布（WebAP サーバ 4 台）

よって、このシステムモデルをベースにして、設定パラメータである WebAP サーバの台数を 2 台から 4 台に変更したモデルの設計を用いることで、顧客要求を満たすシステムを構築することができる。

また、可用性の評価結果を図 16 に示す。最上段の棒グラフは、システム全体の故障率を表し、以降の棒グラフは、Single Point Of Failure（SPOF）及びそれらの故障率を表す。ここで、特に高可用性が求められるシステムにおいては、故障率が大きなロードバランサやスイッチなどの SPOF から優先的に多重化し、SPOF の故障や不具合等によるシステム全体の障害を防ぐ必要がある。

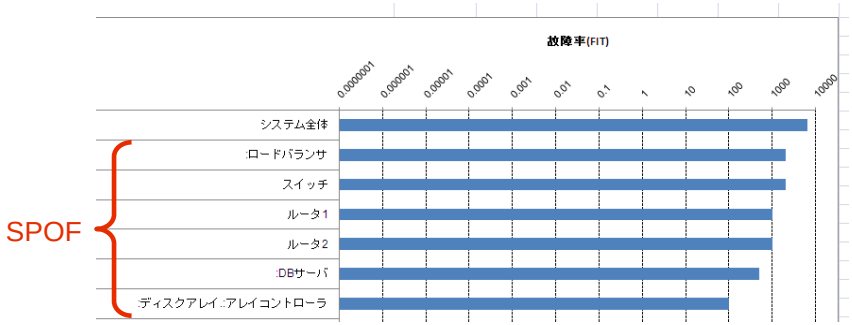

図 16：SPOF 及び故障率

5. 関連技術

本稿のように、拡張プロファイルを定義することでモデリング範囲を限定し、ある分野に特化したモデリングを行うという試みは、様々な分野で行われている。例えば、PFML のベースとして採用した SysML 自体も、主にソフトウェア設計に用いられる UML を、システム設計向けに拡張したものである。また、SPT[5]や MARTE[6]は、リアルタイム組み込みシステムをモデル化し、性能やスケジューラビリティなどの非機能属性の分析を行うために定義された UML の拡張プロファイルである。

しかし、これらの拡張プロファイルは、我々の対象としている IT システムのモデリングに適しているとは言えない。その理由は以下の通りである。

- 比較的幅広い分野をモデリング対象としているため、IT 分野という特定の領域を記述するには、やや一般的すぎる、かつ冗長である。
- IT システムに特有な属性が定義されていない、または不十分である。

そこで我々は、IT システムのモデリングに特化した記法を定義した。PFML では、記述する図の種類を SysML で用意されている図の一部に限定すると共に、後述のような NFR の評価に必要となる種々のプロファイルを定義した。

このように形式的に記述された設計モデルを基に NFR を評価する技術は、特に性能の評価において、数多く提案されている [7)~10)]。文献 7)では、様々な拡張プロファイルを用いて記述された UML モデルや、その性能評価モデル間の連携をサポートする共通インタフェース「PUMA」を定義し、UML で記述された設計モデルを、種々の性能評価モデルに変換している。文献 7)では、この PUMA を使ってセキュリティの側面からモデルの性能分析を行っている。ここでは、SSL によるデータ通信の様子をモデル化し、データの暗号化による通信速度への影響を分析している。文献 9)では、ソフトウェアの性能リスクに関する属性をアノテーションとして付与した UML モデルを性能評価モデルに変換し、ボトルネックとなるリソースやシナリオを算出している。文献 10)では、UML のアクティビティ図を PEPA（Performance Evaluation Process Algebra）ネットモデルに変換している。また、文献 11)12)では、UML や AADL で記述したモデルを動的な故障木（Fault Tree : FT）に変換し、信頼性の分析を行っている。

CASSI によるシステムモデル評価はこれらの技術と近いが、CASSI の特長として、振る舞い図だけではなく構造図にもアノテーションを付与し、性能評価モデル生成時に、それらの割当関係を基に各シーケンスにおける処理時間を算出している点が挙げられる。これにより、アプリケーションの詳細な動作がわからない設計段階でも、大まかな通信データ(メッセージ)のサイズやプラットフォームに依存する固定値から、ある程度の性能（処理時間）を見積もることが可能となる。

また、シミュレーションによる性能評価は、Hyperformix[13]などの性能シミュレータや、それをベースとした性能予測ツール[14][15]などでも用いられている。文献 16)では、性能シミュレーションにより、システムのボトルネック箇所を明らかにした上で、様々な改善策を採用した場合の性能改善率についてもシミュレーションを行い、最適なチューニング方策を導出している。文献 17)では、ソフトウェアやハードウェアの機能レベルでの設計情報をモデル化し、処理時間の見積もりを行っている。これらのツールや技術を用いたシミュレーションを行う場合には、評価者が独自にシミュレーション用のモデルを作成する必要がある。CASSI では、システムの設計書となるシステムモデルの中に、性能評価に必要なプロファイルをあらかじめ記述しておくことで、内部で自動的に性能評価モデルに構築し、シミュレーションを行うことができる。文献 18)19)では、MARTE プロファイルを用いたシステム設計に対する性能検証手法を提案しているが、ここでは、主に組み込み系のシステムを対象としており、IT システムなどのネットワーク化されたシステム全体のモデル化、及び性能シミュレーションは行われていない。CASSI では、IT システムを対象とした評価、つまり、リソース間の通信やネットワークの遅延等も考慮した性能シミュレーションを可能にしている。

6. まとめと今後の課題

本稿では、我々が現在開発中の社内向け MBSI 環境「CASSI」を用いたシステムモデルの性能及び可用性の評価手法について紹介した。CASSI を用いることで、システムの設計段階で顧客要求を評価しておくことで、手戻り工数を削減すると共に、高品質なシステムを提供することができる。また、システムモデルの再利用により、システムの設計工数を大幅に短縮し、SI の生産性を向上させることができる。現在我々は、CASSI を社内向けに公開し、実プロジェクトに適用することで、本手法の有効性について評価している段階である。この評価結果は、完了次第、別途報告したい。

今後の課題は、2.2 節に示した、性能、可用性以外の NFR についても評価を行えるように CASSI を拡張することである。特に、システムの拡張性や機密性の評価に関して、現在検討を行っている。また、システムの構成要素を複数のシステムモデルの中で逐一記述することは、手間がかかりモデル記述者の負担となると共に、記述ミスにつながる。そのため、全てのシステムモデルにおいて共通して利用できる適切なライブラリを用意し、その中から各モデル中に記述する要素を選択できるようにする必要があると考えている。

参考文献

1) A. Avizienis et al, “Basic Concepts and Taxonomy of Dependable and Secure Computing”, IEEE Transactions on dependable and secure computing, Vol. 1, No. 1 (2004)
2) Object Management Group, “OMG Systems Modeling Language” (2006)
3) A. Kobayashi, T. Osaki, T. Nishizawa, ” Benefits of Model-based and Computer-aided System Platform Integration Process for Project Management”, Promac2010 (2010)
4) Jianwen Xiang, Kazuo Yanoo, “Automatic Static Fault Tree Analysis from System Models”, in Proc. of The 16th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'10), FA Track (2010)
5) Object Management Group, “UML profile for Schedulability, Performance, and Time Specification” (2002)
6) Object Management Group, “A UML Profile for MARTE : Modeling and Analysis of Real-Time Embedded systems” (2008)
7) M. Woodside et al, “Performance by Unified Model Analysis (PUMA)” In Proc. 5th Int. Workshop on Software and Performance WOSP'2005 (2005)
8) M. Woodside et al, “Performance Analysis of Security Aspects by Weaving Scenarios Extracted from UML Models”, Journal of Systems and Software, 82(1) (2009)
9) V. Cortellessa et al, “Model-Based Performance Risk Analysis”, IEEE Transactions on software engineering, Vol. 31, No. 1 (2005)
10) C. Canevet et al, “Analysing UML 2.0 activity diagrams in the software performance engineering process”, 4th international workshop on Software and performance (2004)
11) G. J. Pai, J. B. Dugan, “Automatic synthesis of dynamic fault trees from uml system models,” in Prof. of The 13th International Symposium on Software Reliability Engineering (ISSRE’02) (2002)
12) J. Dehlinger, J. B. Dugan, “Analyzing Dynamic Fault Trees Derived From Model-Based System Architectures”, NUCLEAR ENGINEERING AND TECHNOLOGY, Vol.40,No.5 (2008)
13) HyPerformix, http://www.hyperformix.com/
14) 西岡大祐、今木常之、長澤幹夫、“WISE ツールによる情報システム構築支援”、情報処理学会研究報告 EVA [システム評価] (2001)
15) 西岡大祐、山賀晋、長澤幹夫、“システム性能シミュレーションのリモート実行環境の開発”、情報処理学会コンピュータシステムシンポジウム 2001、Vol.2001, No.16 (2001)
16) 河野知行、森本舞、“簡単なシミュレーションによるボトルネック箇所の特定と改善効果の判定”、情報処理学会研究報告 EVA [システム評価] (2002)
17) 関誠司など、“機能・性能シミュレーション連携方式による性能予測手法”、情報処理学会研究報告 EMB [組み込みシステム] (2008)
18) 磯田誠、田村直樹、“UML MARTE Profile を用いた性能シミュレーション手法の提案”、FIT2010（第 9 回情報科学技術フォーラム） (2010)
19) 河原など、MARTE プロファイルを適用した UML モデルによる組込みシステムの性能評価シミュレーション、FIT2010（第 9 回情報科学技術フォーラム） (2010)