情報処理学会論文誌 Vol.57 No.10 2272–2283 (Oct. 2016)

ネットワーク設計書を用いたインフラ品質のためのシステムメトリクスの提案

尾花 将輝[1,a)] 花川 典子[2,b)]

受付日 2015年12月14日, 採録日 2016年7月5日

概要：コンピュータシステムのインフラストラクチャの品質を改善するために，ネットワーク設計書のデザインシートを用いたインフラストラクチャの特徴を計測するメトリクスとその利用方法を紹介する．提案するメトリクスは，インフラストラクチャの規模を示す LOI（Line of Item），結合度を示す CBD（Coupling Between Design sheets）である．LOI はデザインシートの設定項目数とし，CBD はデザインシート間で相互参照されている項目値数とする．大規模プロジェクトに適用して 52 デザインシートのメトリクスを計測した．1 つのデザインシートをコピーして作成した類似デザインシート群の LOI 値はほぼ等しく，さらに CBD 値が高いことが分かった．また，障害レポートの障害原因となったデザインシートは，非常に LOI 値が大きいという特徴と，LOI 値が小さく，かつ，CBD 値が高いという特徴が確認できた．LOI 値と CBD 値を改善するために，共通デザインシートを生成し，値の変更や項目の追加等の差分のみをデザインシートにする方法を示した．コピーして作られた 25 個のデザインシートでは LOI 値と CBD 値が大きく減少し，障害発生の原因であった大規模デザインシートを 30 個のデザインシートと共通デザインシートに分割することで，LOI 値の 9 割近くを削減することができた．今後はコンフィグレーションファイルの規模と内容とデザインシートとの関係を追及する予定である．

キーワード：インフラストラクチャ，メトリクス，ネットワークプロダクト品質，規模，結合度，デザインシート

System Metrics for Infrastructure Based on Network Design Sheets

Masaki Obana[1,a)] Noriko Hanakawa[2,b)]

Received: December 14, 2015, Accepted: July 5, 2016

Abstract: We propose infrastructure metrics and usefulness of the metrics to improve infrastructure quality. The metrics are LOI (line of item) and CBD (Coupling Between Design sheet). LOI means the number of items that are set in design sheets, CBD is the number of items that are referred from the other design sheets. The metrics were applied an industrial large-scale computer system including infrastructure. As a result, we confirmed that the copied design sheets have higher values of LOI and CBD. In addition, we confirmed that a design sheet which caused system failures has a highest value of LOI. In contrast, if a sheet has a little value of LOI and a high value of CBD, the sheet might cause system failures. In future, we will provide relationship between configuration files and design sheets.

Keywords: infrastructure, metrics, network product quality, scale, coupling, design sheet

1. はじめに

近年，公共交通システム，金融システム，証券取引システム等の重要性は増加している．特に東京証券取引所システム等の金融関係のシステムダウンは国内だけでなく，世

1 大阪工業大学
Osaka Institute of Technology, Hirakata, Osaka 573–0196, Japan
2 阪南大学
Hannan Uniersity, Matsubara, Osaka 580–8502, Japan
a) obana@is.oit.ac.jp
b) hanakawa@hannan-u.ac.jp

界規模での混乱を招く．また，大規模システムはモバイル端末や通信技術を含む高機能化や高性能化が要求され，ソフトウェアだけでなく，インフラストラクチャも大規模化，複雑化の一途をたどっている．インフラストラクチャ（以下インフラ）とは，コンピュータシステムのアプリケーションソフトウェア以外の部分を示し，サーバのハードウェア構築や，そのOS，ミドルウェアの設定，ネットワーク関係の機器の設定，仮想化技術で構築された仮想マシンの設定等で構成される．インフラ構築の複雑さが影響し，近年では社会的に影響を与えるシステムダウンに直接結び付くケースも多数存在する．

たとえば，2013年1月から6月の日本の新聞紙上で取り上げられた社会に影響を与えたコンピュータシステム障害は21件であった．純粋なソフトウェア障害が起因したシステムダウンは6件であり，インフラ構築の設定ミスによるシステムダウンが11件，物理的機器故障等によるシステムダウンは4件であった [1]．このように，大規模化，複雑化するインフラ構築における機器の設定ミスが，コンピュータシステム全体のシステムダウンを引き起こす重要な要因の1つとなっている．本稿で検証対象としたプロジェクトは，サーバ，クライアント，ネットワーク機器等，総数1,471台のインフラ構築である．52個の表計算ファイルでインフラ設計情報が管理され，そのワークシート数は929個，設定すべき項目数は224,907であった．同じ値を複数ワークシートに記入された箇所も多く，1つの設定項目値の変更が発生すると，すべてのワークシートを精査する必要があり，その作業は非常に煩雑であり，修正抜け等のミスを誘発する可能性が高い．

そこで，本稿ではコンピュータシステムにおけるインフラの規模，結合度を計測するメトリクスを提案する．著者らは文献 [2], [3] で基本的なアイディアの概要を提案し，文献 [4] で初期メトリクスを提案し，その後実験を追加して精査した．その結果に基づいて，新メトリクス提案とその適用事例を本稿で述べる．ソフトウェアと同様に「計測できないものはコントロールできない」の概念 [5] を基に，インフラの規模と結合度を計測し，その値に基づいてインフラの品質を管理，改善することを目指す．従来のインフラの品質の研究はハードウェア故障時のリスク等が主であった [6], [7]．本研究では，ハードウェアの物理的故障は範囲外とし，インフラ構築時の設定ミス等に着目する．設定ミスは設計上の不良やヒューマンエラーが要因と考えられる．ヒューマンエラーであるならば，設計書の規模の大きさや結合度の高さがエラーを誘発する可能性は高い．そこでインフラの規模と結合度をネットワーク設計書を用いて定量的に計測する．これによって，大規模なインフラの規模や構造を明らかにし，品質向上に役立てることを目指す．

2章では関連研究を示し，3章でインフラの規模，結合度のメトリクスを提案する．4章で実際のコンピュータシステムのインフラで計測する．計測した適用事例をもとにコピーして作成したデザインシートと障害の発生したデザインシートについて5章で述べる．6章で計測されたLOI値とCBD値の改善方法を提示し，7章でまとめと今後の課題について述べる．

2. 関連研究

多くのインフラやシステムアーキテクチャの設計技術が提案されている．Clementsらは ATAM（Architecture Tradeoff Analysis Method）というインフラの設計手法を提案した [8]．ATAMは非機能要求を技術的要求へ変換するためのテンプレートと，インフラの非機能要求と技術要求のトレードオフで設計する方法を提供した．また，Hasegawaらは早い技術革新に対応したインフラ設計方法を提案した [9]．特徴は，ビジネス分析と同時に実施される反復インフラ設計プロセスである．これらの設計方法はインフラの設計に有効であるが，品質を管理するための指標等は提案されていない．我々の提案するメトリクスはインフラの規模と結合度等を定量的に計測する．計測された値と品質との関係を明確にすることで，品質予測への貢献が期待できる．

また，クラウンドコンピューティングサービスのためのインフラ設計方法も提案されている [10]．インフラのコストと，サービスレベルの合意違反を発生させるビジネスロスをトレードオフして，サーバ数やルータ数，通信帯域幅を決定する方法である．Touziらは異なる組織での協調サービスのための Model-Driven Architecture（MDA）に基づくアーキテクチャ設計方法を提案した [11]．これらのインフラ設計方法は様々なサービスとのトレードオフが含まれており，サービスはソフトウェアだけではなく，ハードウェアを含むインフラも対象となる．したがって，ソフトウェアとハードウェアのパッケージをサービスと見なしている．我々の研究では，ソフトウェア開発とインフラ構築は異なる作業プロセスで，異なるスケジュール上で進むと考える．ソフトウェア技術者はソフトウェアを開発し，インフラ技術者はインフラを構築する．我々のメトリクスは実際の産業界の開発プロセスに近い考え方を取り入れた概念である．

さらに，インフラを構築するときのヒューマンエラーによる設定ミスに関する研究も行われている．Pappasらは DNSのヒューマンエラーによるコンフィグファイル設定ミスについて調査した [12]．DNSコンフィグファイルの設定ミスは DNSゾーンの15%エリアに影響を及ぼし，DNSゾーンの21%に設定の矛盾が生じていることを明らかにした．大規模システムのインフラのコンフィグファイルの単純ミスを防ぐための自動チェックシステムの必要性を主張した．また，Woolは Firewallコンフィグファイルの設定ミスによるエラーを定量的に計測した [13]．システムが

大規模になると，コンフィグファイル設定ミスは劇的に増加すると示唆する．これらの研究のコンフィグファイルのエラーを量的に計測する点で我々の研究に類似する．しかし，これらの研究では DNS や Fire Wall という特定の機器に依存しているが，我々の研究ターゲットはインフラすべてを含む点で異なる．

一方，コンピュータシステムの高品質，安定稼動のために，システム規模や重要度別に考慮すべき事項が定義された非機能要求グレードが存在する [14], [15]．非機能要求グレードとは，ユーザ業務をサポートする具体的な機能以外の，インフラも含むシステム全体の可用性，運用・保守性，移行性，セキュリティ，システム環境等を定量的に要件定義するための手法である．社会的影響度の大きさ（重要度）やシステム規模によってそれぞれ，非機能要求を定義できる．日本では IPA（情報処理推進機構）から非機能要求グレード表が提供され，システム要件定義時に有効活用されている．本稿の提案メトリクスの目的は，非機能要求グレードの目的と類似する．特に非機能要グレードの「運用保守性」の「障害時運用」の「復旧作業」,「障害時の運用管理方針」の「問題管理」,「構成管理」,「変更管理」に本メトリクスは関連する．定性的に定義された非機能要求グレードを提案メトリクスで定量的に補足できる．たとえば，非機能要求グレードの「障害運用」の「復旧作業」の具体的な対象は機器の物理的故障であり，かつ，定性的な定義である．しかし，「1. はじめに」で述べたように，機器の設定項目の設定ミスが原因の障害も発生している．障害時の復旧作業の定義の一部に機器故障以外の設定ミスの箇所特定のための項目を追加するならば，機器故障以外の設定ミスによる障害も考慮することができる．また，「障害時の運用管理方針」の「問題管理」プロセスを定義するときにも，本メトリクスを用いて設定ミスによる障害の発見を補足することが可能となる．

3. インフラストラクチャメトリクスの提案

3.1 インフラ構築とは

本稿で利用する「インフラ構築」について説明する．インフラストラクチャとはコンピュータシステム上の機器，OS，ミドルウェアであり，それらを構築する作業をインフラ構築とする．機器はコンピュータ機器，ネットワーク機器，周辺機器に分類される．コンピュータ機器はサーバコンピュータ，クライアントコンピュータ，モバイルターミナルを含み，ネットワーク機器はルータ，アクセスポイント，負荷分散機器，スイッチングハブを含む，周辺機器はプリンタ，スキャナ，UPS やそのほかの様々な最新機器も含む．

また，ミドルウェアとはデータベースサーバ，Web サーバ，ファイルサーバ，メールサーバ，プロキシサーバ，DNS サーバ，認証サーバ，VPN サーバ，ログ監視サーバ，バックアップサーバ，セキュリティサーバ，Fire wall サーバ，アプリケーションサーバ等である．サーバコンピュータとクライアントコンピュータ，モバイルターミナルは Linux や Windows，Android 等の OS が装備される．サーバソフトウェアや OS はそれぞれのバージョンやディストリビューションを持ち，それぞれのバージョンで機能や設定項目が少しずつ異なる．

インフラ構築作業は，機器搬入，機器設置，機器セットアップ，OS とミドルウェアインストールとセットアップと環境整備作業，最終動作確認作業等である．近年はインフラの高性能化，高機能化にともない，物理機器だけでなく仮想機器の種類や数が増加して，ハードウェアとサーバソフトウェアのセットアップ項目数は劇的に増加している．さらに，モバイルターミナルの導入はクライアント数を劇的に増加させ，端末ごとに異なる OS はセットアップ作業量の大幅な増加につながる．

これらのインフラ構築の手順はソフトウェア開発と同様に，要求分析，概要設計，詳細設計，構築，テスト，総合テストのフェーズに分かれている [16]．インフラ構築手順の詳細な定義は本稿のテーマ対象外とする．本稿では産業界で使われている実践的な開発手順を使用して議論する．

インフラ構築のドキュメントは，概要設計書，操作設計書，セキュリティ設計書，ネットワーク構成設計書，運用設計書等がある．**図 1** にネットワーク構成の TCP/IP 相関図と**図 2** にハードウェアの概要設計のサンプルを示す．詳細設計フェーズでは，各機器の具体的な設定値を設計するためのデザインシートがある．**図 3** は図 2 の仮想マシンのデザインシートの一部である．ただし，記載されている値は実際の値ではなく，機密のためにコンピュータ名を「name047」，IP アドレスを「address094」等，意味のない値に置き換えたデザインシート例である．仮想マシンは物理的機器ではないが，現在のインフラ構築では重要な要素の 1 つである．デザインシートはメモリサイズ，ビデオ

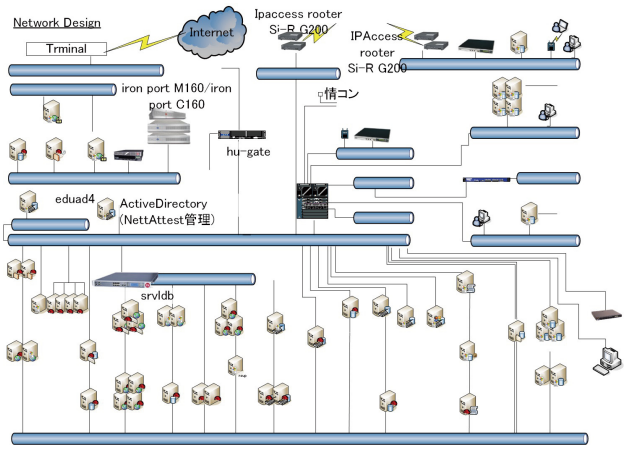


図 1 ネットワーク構成図の TCP/IP 相関図の一部
Fig. 1 A part of a network configuration diagram.

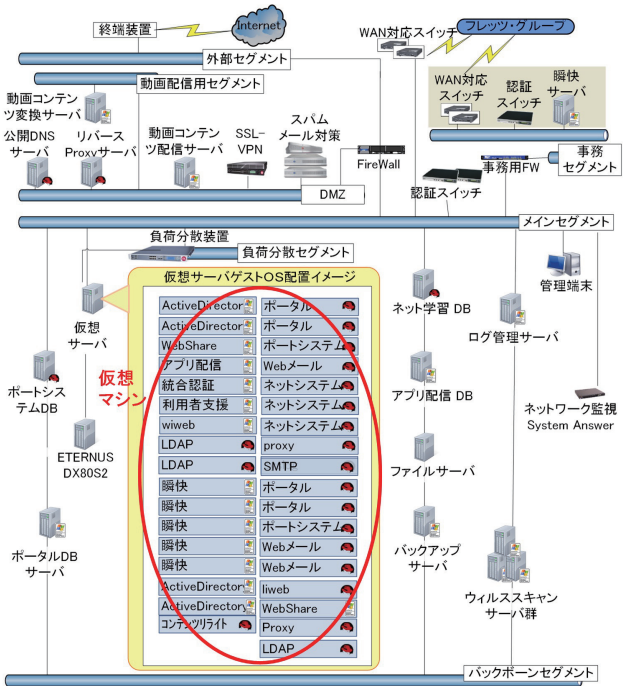


図 2　ハードウェア設計の概要図

Fig. 2　Schematic diagram of hardware design.

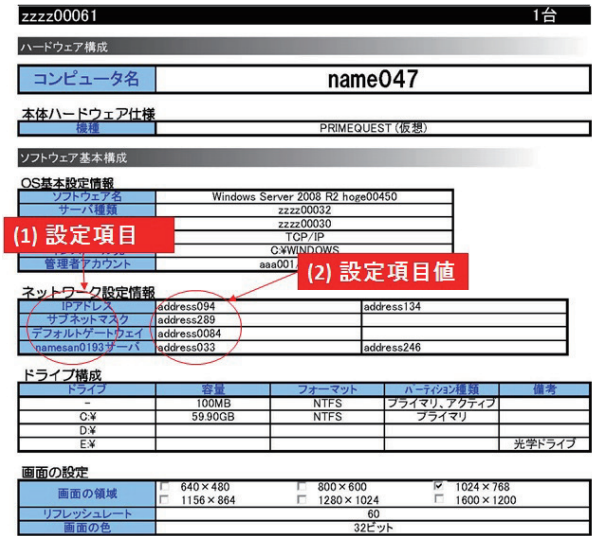


図 3　仮想マシンのデザインシート例

Fig. 3　An example of virtual machine design sheet.

カード，VMCI デバイスの利用，ハードディスク，ネットワーク等，仮想マシンに設定すべき項目がすべて記載される．また，物理的設定値だけでなく，OS やミドルウェア等のインストール時のオプション値も記載される．図 3 の仮想マシンデザインシートの設定される項目の総数は 2,642 個である．システム全体で 10 個の仮想マシンが構築された場合，総設定項目数は 26,420 個となる．このように，インフラ構築の詳細設計に作成されるデザインシートの項目数はインフラの規模の拡大にともない，劇的に増加する．

本稿では，デザインシートの設定すべき項目のことを「設定項目」（図 3 の (1) 参照）とし，その設定項目に入力された値を「設定項目値」（図 3 の (2) 参照）と呼び，それぞれを区別する．

3.2　デザインシートとコンフィグファイルの関係

デザインシートとはインフラ構築の際に生成されるドキュメントの中で特にネットワーク機器の設定値や，OS の設定値等が詳細に記載されたものを指す．デザインシートは基本的に機器ごとに作成されるが，便宜上，複数の機器をまとめて建屋やフロアごとにデザインシートが作成されることもある．これを一般企業では詳細設計書等と呼ぶこともあるが，本稿ではデザインシートと呼ぶ．デザインシートは機器ごとにフォーマットが異なるが，その機器に必要な情報すべてが記載される．

本稿で対象とするデザインシートは，1 つの MS-Excel のファイルに複数のワークシート，設定項目が含まれる．デザインシート分割方法，ワークシートの分割方法や記載される項目はそれぞれの組織の規則や過去の経験等に従う．機器ごとのデザインシートだけでなく，建屋ごとのデザインシートの場合もある．ワークシートも設計者の理解しやすさを優先したわけ方である．ただし，いかなる分割方法であったとしても，機器名，設置場所，パラメータ値等が機器設定にかかわる項目が詳細に記載されている．

たとえば，コアスイッチ等の場合，1 つの MS-Excel ファイルで管理されており，中には機器名，設置場所，どのポートがどのフロアのどの機器に接続し，どのような VLAN（仮想 LAN）を利用させて，その VLAN はどのような IP（DNS 名も含む）が利用できるかが詳細に記載されている．

他にも，各建屋で設置されるインテリジェントスイッチ等が，どの LAN ポートがどのようなフロアにどの VLAN で接続するか等が詳細に記載されている．その際にはプリンタやスキャナ等のネットワークが接続されている機器も詳細に管理されている．

開発者はこれらの情報をもとに設置される機器を設定する．設定する際には機器や OS 等によって設定方法は異なる．たとえば，ネットワーク機器等のコンフィグファイルはコンフィグ設定専用ツール等のソフトウェアである程度は自動的に生成される．システム特有の設定値（たとえばドメイン名や IP アドレス等）だけを設定ツール上で入力するか，生成されたコンフィグファイルを直接編集する．その際に，デザインシートを参照し，コンフィグ設定ツール上でチェックボックスにチェックを入れたり，ドメイン名等を直接入力したりする．そしてコンフィグ設定ツールが自動的にコンフィグファイル生成する．つまり，デザインシートは設計段階でインフラ全体を設計するときに作成され，その値に従ってコンフィグファイルが自動，または手動で作成される．デザインシートに誤りがあれば，コンフィグファイルも誤った設定になり，結果，ネットワーク障害になることがある．

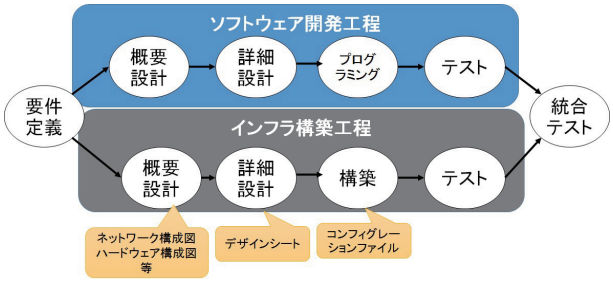


図 4 ソフトウェアとインフラストラクチャ構築プロセス
Fig. 4 Processes of software development and infrastructure construction.

3.3 ソフトウェア開発プロセスとインフラ構築プロセス

図 4 にソフトウェア開発プロセスとインフラ構築プロセスを示す．要求分析フェーズでは，ソフトウェア開発技術者とインフラ構築技術者と顧客がともに要件を決定する．その後，ソフトウェア開発技術者はソフトウェア開発，インフラ構築技術者はインフラ構築をそれぞれ独立したプロセスとスケジュールで並行に実施する．2 つのプロセスの外観は類似しており，概要設計，詳細設計，実装，テストとなる，その後，システム全体を統合して総合テストが行われる．インフラ構築の詳細設計フェーズでは図 3 で示すデザインシートを作成し，実装フェーズではデザインシートを基にコンフィグファイルを生成するためのシェルやバッチファイルを作成し，実行する．また，デザインシートを参照しながら，設定専用ツールを使用して手動で入力したり，コンフィグファイルを直接編集したりすることもある．

本稿で提案するインフラの規模 LOI（Line Of Items）と結合度 CBD（Coupling Between Design sheet）は，詳細設計フェーズで作成されるデザインシートを計測対象とする．本来は構築フェーズで作成されるコンフィグファイルを対象とすべきであるが，コンフィグファイルはデザインシートから自動生成したり，メーカから提供されたコンフィグファイルの一部を変更するのみの作業となったりするケースが多い，メーカや機器に依存するコンフィグファイルとなり，汎用性が下がるので，デザインシートを計測の対象とした．

3.4 計測するメトリクス

以下の 2 つのメトリクスを提案する．

- LOI：デザインシートの規模（設定項目数の総数）
- CBD：デザインシートの結合度

3.4.1 LOI（Line of Item）

インフラの規模はデザインシートの LOI で計測する．以下の式で計算する．

$$LOI_i = num_i \tag{1}$$

num_i：i 番目のデザインシートの設定項目数

図 3 の表形式のデザインシートの実際に入力する項目の数をカウントする．入力する項目数が多いと，インフラの規模が大きい．さらに，規模が大きければ対象プロダクトは複雑であると考え，ヒューマンエラーが発生しやすいと考える．

3.4.2 CBD（Coupling Between Design sheet）

デザインシート間の結合の強さを CBD_i で計測する．以下の式で計算する．

$$CBD_i = \sum_{j=0}^{m}\left(\sum_{k=1}^{a}(P_{ik} * P_{jk})\right) \tag{2}$$

P_{ik}：k 番目のシステム固有値が i 番目のデザインシートにあるとき　1
　k 番目のシステム固有値が i 番目のデザインシートにないとき　0

P_{jk}：k 番目のシステム固有値が j 番目のデザインシートにあるとき　1
　k 番目のシステム固有値が j 番目のデザインシートにないとき　0

m：デザインシートの総数

a：システム固有値の総数

デザインシート間の結合を計測する．たとえば，IP アドレス 150.168.1.1 がデザインシート A とデザインシート B にそれぞれ記入されていると，デザインシート A とデザインシート B に結合関係があると考え，結合度は 1 となる．さらに，別の IP アドレスが A と B のデザインシートに記入されていると，その結合度の値は 2 となる．結合度が強いと対象プロダクトは複雑であると考え，ヒューマンエラーが発生しやすいと考える．たとえば，上記の場合，デザインシート A の 150.168.1.1 を 150.168.1.2 に変更した場合，必ずデザインシート B の 150.168.1.1 も 150.168.1.2 へ変更する必要がある．しかし，デザインシート B の修正を忘れるとヒューマンエラーによる障害が発生する．修正ミスの可能性が高くなる箇所を示唆するためのメトリクスである．

CBD の計測対象となる設定項目は，基本的に設定項目値が自由記述で書かれている IP アドレスや DNS 登録名のようなシステム固有の値とする．たとえば，図 3 の仮想マシンデザインシートでは，先頭の「コンピュータ名」の「name047」（実際にはコンピュータ名称が記載されるが機密のため無意味な文字列に置き換えた），ネットワーク設定情報の「IP アドレス」の「address094」の値等である．ドライブ構成の容量の欄の「100 MB」やフォーマット欄の「NTFS」の値はシステム固有の名称ではなく，一般的に使われる容量単位やファイルシステム名称なので CBD の対象とはしない．もちろん，「ON」や「OFF」等の設定値も対象としない．システム固有値は，あらかじめ「CBD 対象システム固有値リスト」を作成して利用する．このリストはシステム設計で利用される IP アドレス一覧，DNS 登

録名一覧等を利用してシステム固有値リストを作成する．

4. メトリクスの計測

4.1 ターゲットプロジェクト

提案するメトリクスを産業界のプロジェクトに適用する．ターゲットプロジェクトは，様々なアプリケーション，ミドルウェア，大規模インフラを含むシステムである（**表 1** 参照）．開発期間は 2011 年 10 月から 2012 年 3 月，クライアント PC 数は 1,184，物理サーバ数は 19，仮想マシン数は 35，パッケージソフトを含むサーバソフトウェア数は 7，ネットワーク機器数は 258，クライアント側の主なアプリケーション数は 36 のコンピュータシステムである．

4.2 デザインシート

ターゲットプロジェクトの詳細設計フェーズでは 52 個のデザインシートが作成された．デザインシートは MS-Excel 等のスプレットシートソフトで作成され，他のプロジェクトでも利用できる汎用性を高める工夫をしている．たとえば，図 3 に示すように，仮想マシン設定用のデザインシートを用意する．「コンピュータ名」や「ソフトウェア名」等の設定項目はあらかじめ設定されており，具体的な設定項目値を空欄とするデザインシートである．設計するシステムによって追加項目や項目の構成を変更することもある．一種の雛形やテンプレートである．他プロジェクトで仮想マシンを設計するときに汎用的に利用できるものがデザインシートである．ただし，汎用デザインシートをもとに作成したデザインシートだけではなく，ターゲットプロジェクト専用に作成した独自のデザインシートもターゲットプロジェクトに含まれる．

表 1 検証対象のプロジェクトの概要
Table 1 An outline of the target project.

開発期間	クライアント台数	サーバ台数	仮想マシン数	サーバソフト数	ネットワーク機器数	クライアントソフト数
6 カ月	1,184	19	35	7	258	36

設計担当者はデザインシートの設定項目に設定項目値を記入する作業となる．チェックボックスにチェックを入れる場合や，IP アドレスやポート番号を数値で入力，命令とパラメータで構成されるコマンドラインをテキストで入力する場合もある．また，サービスリストや IP アドレスリスト等のリスト関係も手作業で作成する必要がある．

デザインシートは複数のサブシートを持つ．スプレットシートソフトの MS-Excel の例では，Book ファイルが 1 つのデザインシートとなり，その中のワークシートをサブシートと呼ぶ．サブシートに分ける理由は，1 つのデザインシートが大きくなりすぎて，意味ある塊にわけたサブシートを作って管理をしやすくするためである．たとえば，図 3 で示す仮想マシンのデザインシートでは 14 のサブシートを含む．サブシートは（1）OS 設定，（2）kernel パラメータ，（3）インストールオプション，（4）ネットワーク設定等，意味ある項目がサブシートに分類されている．

デザインシートは主に物理機器や仮想機器ごとのようにサーバ機能（サービス）で作成される．たとえば，ファイルサーバのようなストレージ機器は 1 つのデザインシートを作成する．仮想マシンも物理マシンと同様に，1 つのデザインシートを作成する．ただし，環境設定に利用する IP アドレス一覧やサービスパラメータ一覧等のデザインシートは対象機器が存在しない場合もある．また，便宜上，建屋ごとにデザインシートを作成する場合もある．

ターゲットプロジェクトの詳細設計では，929 個のサブシートを含む 52 個のデザインシートを作成した．

4.3 計測結果

図 5 に計測した結果を示す．LOI の最大値は 29,700 であり，1 つのデザインシートに 29,700 個の設定項目があるという意味になる．最小値の LOI は 40 であった．本システムのインフラの総 LOI は 224,907 となる．CBD は式 (2) に基づき計算し，52 デザインシートを CBD 値の降順に並べた結果である．最大の CBD 値は 24,018 であり，最小値は 122 である．LOI ではデザインシートの規模の目安

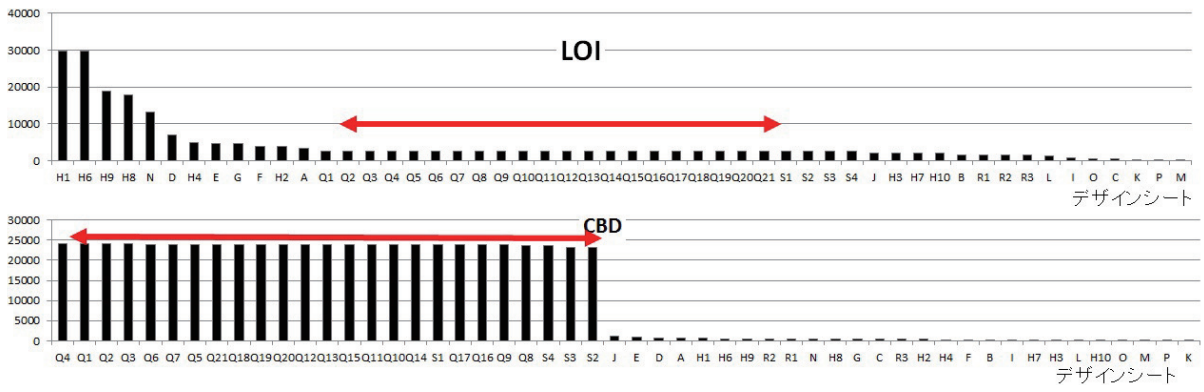


図 5 LOI，CBD の計測結果
Fig. 5 Measurement results of LOI and CBD.

となったが，CBD 値では極端に大きい値のグループとそれ以外に分かれ，単純に比較するには難しい．次章で，計測値の活用方法の例を示す．

5. 提案メトリクスの活用事例

5.1 コピーして作成した冗長性の高いデザインシート

ターゲットプロジェクトのデザインシートの中に，1 つのデザインシートをコピーして類似のデザインシートを作成したケースがあった．基本となるデザインシートを 1 つ作成して，それを複写して類似のデザインシートを作成する方法は作業の効率化のために頻繁に実施される．コピーしたデザインシートで同一設定値はそのまま流用し，異なる部分だけを変更して利用する方式である．ターゲットプロジェクトでは，21 個の仮想マシンデザインシートと 4 個の物理マシンデザインシートがそれぞれコピーして作成された．本プロジェクトでの仮想マシンのホストは VMware vSphere 5.0 を使い（本ホストの設定も 2 つのホスト OS 用デザインシートで設定された），21 台の仮想マシンが設定された．仮想マシンでは各種アプリケーションの Web サーバやファイルサーバ等が設定される．また，仮想マシンではない物理的に独立したサーバは 4 台あり，それが 4 個の物理マシンデザインシートとして作成された．仮想マシンと物理マシンの差はあるが，基本的にはアプリケーションサーバの設定内容が記載されたデザインシートである．

そこで，他のデザインシート 27 個と仮想マシンと物理マシンのコピーして作成したデザインシート 25 個の 2 つのメトリクス値の比較をした．図 5 にその結果を示す．矢印で示すように，LOI のグラフでは左から 13 番目から 38 番目までのデザインシート，CBD のグラフでは先頭から 25 番目までのデザインシートがコピーで作成されたデザインシートである．さらに，LOI，CBD を 2 軸に持ち，52 デザインシートを 52 個の点とする散布図の**図 6** を作成した．コピーして作成したデザインシートは点線で囲った範囲となり，明らかに独自のグループを形成していることが分かった．

インフラ設計では，各種アプリケーションサーバはそれぞれ独自のデザインシートを作成する場合が多い．つまり，異なるアプリケーションサーバは独自の設定項目や重要性の高い項目の違い等の差で，それぞれのアプリケーションサーバ専用のデザインシートを作成するケースである．しかし，本プロジェクトでは，アプリケーションサーバを基本となる 1 つのデザインシートをもとにすべてのアプリケーションサーバを定義した．したがって，対象となるアプリケーションサーバに必要でない設定項目も含まれた冗長なデザインシートとなった．本来不要なはずの設定項目値がそのまま残った箇所も存在した．図 5 や図 6 に示すように，他のデザインシートと比較して CBD 値が極端に高い値を示した．

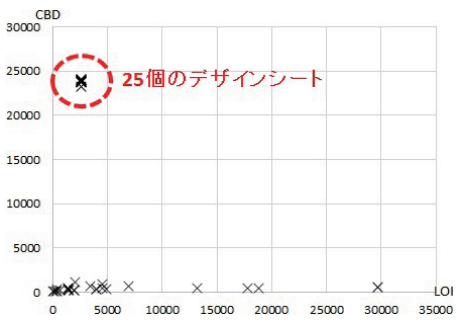


図 6 LOI と CBD の計測結果の散布図
Fig. 6 scatter diagram of LOI and CBD.

5.2 障害との関係

本システムでは，リリース後に 34 件のインフラに関連する障害が報告された．障害はすべて解決されており，その原因は解明されている．34 件の障害のうち，設計段階の作業や作成したデザインシートと関係のある障害の 2 件に着目した．それぞれの障害とその原因，それに関係するデザインシートの記載上の問題等を以下に示す．

5.2.1 障害 NO.1：IP アドレス重複設定障害

本システムの稼動後に，フロア A の PC よりプリンタが出力できないという問題が発生した．フロア A では PC が 30 台設置されており，フロア A に設置した専用プリンタに出力する設定である．しかし，PC からプリンタ出力ができないという障害が発生した．

原因を調査した結果，フロア A 内で利用できる無線 LAN で接続されるノート PC の台数が予想を上回った．結果，プリンタが利用する IP アドレスまで浸食してしまい，ノート PC とプリンタの IP アドレスが重複してしまうという問題が発生した．これらの対策として，プリンタの IP アドレスを例外としてノート PC 等に振らないようにすることで，障害の対応を行った．

この問題について，デザインシートを分析した．H1 のデザインシートでプリンタのネットワーク設定が管理されていた．H1 はフロア A がある建物のネットワークを管理しているデザインシートであり，建屋に含まれる複数個のアクセススイッチの設計が記載されており，建屋内にある非常に多くの VLAN や，DNS 名等が記載されていたので，H1 デザインシートの LOI 値が本システムの最大値となった．

5.2.2 障害 NO.2：認証キー問題

障害 NO.2 は，ファイルサーバへブラウザからアクセスする際にエラーが発生し，接続できない障害である．原因はファイルサーバへアクセスする際にブラウザから SSL 通信を用いて行われるが，その SSL 通信に利用する RSA 証明書のキーの長さが短いためにクライアントが認証をエラーとし，接続できなかった．クライアント PC の OS がアップデート時にキーの長さに制限がかかったため，それ

以前の設定値ではエラーとなった．

この障害に関するデザインシート C を調査したが，デザインシート C 内に RSA 証明書に関する記述がなく，どのようなキーが利用されているかが定かではなかった．もし，デザインシート C に明確にキーが記述していれば，セキュリティ的に問題のある脆弱なキーであることをレビュー等で見つけることができたかもしれない．さらに OS のアップデートによる認証キーの長さの変更時には，再度デザインシートの認証キー項目を確認することができる．

障害 NO.2 は，必要な設定項目がデザインシート C に欠落していたことに起因する．さらに詳細に原因となったデザインシート C を調査すると，必要な設定項目が多く欠落した内容であり，不十分な記載内容を確認した．設定項目数は LOI 値に影響を与え，欠落設定項目がある場合は LOI 値が低くなる．たとえば，類似の値のデザインシート K は LOI 値が 316 であるが，内容は別棟の建物に設置されるアクセススイッチであり，1 フロアでごく少数の機器を設置しただけだったデザインシートであった．これに対し，デザインシート C の LOI 値は 393 であり，機能や規模に対して小さすぎる LOI 値を示す問題のあるデザインシートであった．

5.3 得られた教訓

障害 NO.1 では，建屋ごとにネットワーク情報がまとめられた巨大なデザインシートが問題と考えられる．従来は建屋ごとにデザインシートをまとめると物理的な区別が明確になり分かりやすかった．しかし，今回対象となった建物は，最新のネットワーク設備を集中的に実現する建屋を目指した．そのために非常に多くのネットワーク機器が含まれるデザインシート H1 となった．多くの記入項目が存在し，見落とした可能性が高いと考えられる．本来，プリンタの IP アドレスと DHCP から割り振られるノートパソコン等の IP アドレスが被らないようにプリンタの IP には静的 IP を振るか，例外に追加する必要がある．しかし，この建屋はフロア数も多く，設置機器が多いため，非常に設定が複雑となる．さらに，設定には不要な項目も多く記載されていた．不要な項目とは他のプロジェクトや機器のシートの使いまわしのために本来不要だがそのまま削除しなかった設定項目のことである．結果として，設定項目数が非常に多く設定が複雑になって，ミスに気付かなかったことが原因であると考える．つまり，設定項目に矛盾が発生していてもデザインシート H1 の規模が大きいため，その矛盾に気づかなかった可能性が高いと考える．以下にまとめを示す．

教訓 1 極端に LOI 値の高いデザインシートはミスが潜む可能性がある．

教訓 2 極端に LOI 値の高いデザインシートには不要項目が含まれており，その項目でさらに複雑さが増す．不要項目は適宜，削除をする．

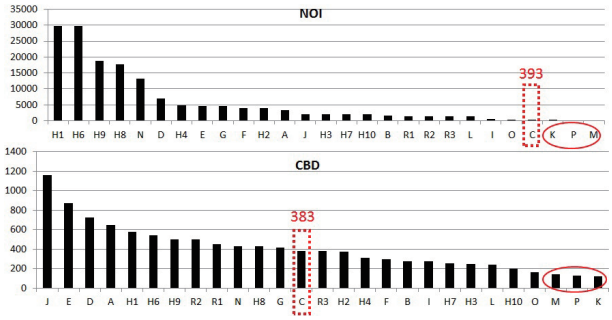


図 7 コピーして作成したデザインシートを除外した計測結果
Fig. 7 Metrics values excluding copied design sheets.

教訓 3 建屋ごとのデザインシートでは極端に LOI 値の大きなデザインシートが発生しやすい．従来のデザインシート形式にとらわれず，最適なデザインシートの分割方法を考えることが重要である．

続いて障害 NO.2 の認証キーの問題は，障害 NO.1 とは逆に設定項目数が少なすぎたデザインシート C の問題と考える．障害 NO.2 のデザインシート C の LOI 値は 393 であり，CBD 値は 383 である．**図 7** は極端に値の大きい 25 個のコピーして作成されたデザインシートを除外した LOI 値と CBD 値の計測結果である．デザインシート C の LOI 値は最小から 4 番目であるにもかかわらず，CBD 値はデザインシート中の中央位置あたり（最大から 13 番目，最小から 15 番目）に存在する．デザインシート C と同等程度のデザインシート K，P，M は LOI 値が最小であると同時に CBD 値も最小である．つまり，デザインシート C だけが他と異なるパターンを示す．

デザインシート C は，他の機器等との関係がある重要な機器にもかかわらず，本来記載されないとならない情報が記載されていないデザインシートという判断もできる．もし，他の機器とも関連が深い（CBD 値が大きい）が，設定項目に欠落が多いということがデザインシート C の特徴であると，今後の障害発生の可能性も考えられるデザインシート（機器）と注目することもできる．以下にまとめを記す．

教訓 4 同規模の LOI 値の他デザインシートと比較して，CBD 値が極端に大きい場合は，必要な項目が欠落しているデザインシートの可能性があるので，欠落項目の有無を確認すること．

このように 2 つのケーススタディではあるが，問題のあるデザインシートの LOI 値や CBD 値の特徴が明らかにできた．本事例が汎用性を持つことが確認できれば，問題のあるデザインシートを設計段階で指摘できる手段の 1 つになると期待できる．

6. メトリクス値の改善方法

本章では提案するメトリクス値を改善する方法を提案

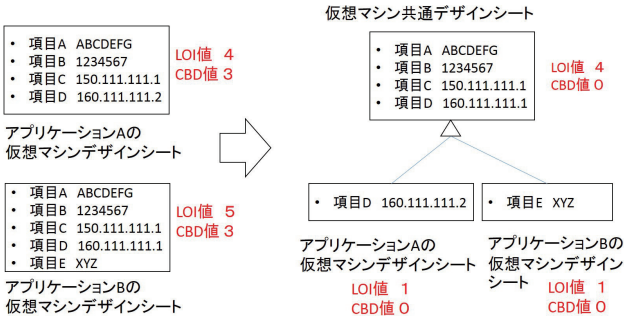


図 8 コピーしたデザインシートのメトリクス値を改善する方法の概要

Fig. 8 Concept of improving the metrics' values for copied design sheets.

し，その方法をコピーして作成した 25 個のデザインシートと障害の発生した規模の大きなデザインシートへ適用した結果を紹介する．

6.1 汎化を利用したメトリクス値の改善方法

図 5 や図 6 に示すように，極端に LOI 値と CBD 値が高いデザインシートが存在する．5.1 節で説明したように，これらは 1 つのデザインシートをコピーして作成した冗長性の高いデザインシート群である．異なるアプリケーションサーバの設定であるが，共通部分と非共通部分の設定項目に別れ，さらに，1 つのデザインシートをコピーして作成したので，当該アプリケーションサーバには不要な項目も含まれた．したがって，設定項目数である LOI 値が上昇し，かつ，本来不要であるはずの設定項目をコピーしてそのまま放置したので CBD 値も上昇した．

そこで，コピーして作成されたデザインシートのように，多くの共通する設定項目と設定項目値があるが，一部の異なる設定項目と設定項目値を含むデザインシートのメトリクス値の改善方法を示す．基本的な考え方は，オブジェクト指向開発方法論の継承概念を適用し，共通部分を汎化して 1 つの上位デザインシートを作成し，各アプリケーションサーバの差分をそれぞれの下位デザインシートとして設定する方法である（**図 8** 参照）．

図 8 の左側には，現在のデザインシート例を示す．アプリケーション A とアプリケーション B のデザインシートの LOI 値はそれぞれ，4 と 5，CBD 値は両者とも 3 を示す．しかし，アプリケーション A とアプリケーション B のデザインシートは項目 A，項目 B，項目 C の 3 つの共通項目を持つ．したがって LOI 値，CBD 値ともに大きくなった．そこで，汎化の概念を導入し，新しく共通デザインシートを設定した（図 8 の右側参照）．ここには共通になる項目とその設定値を定義する．図 8 の例では項目 A，項目 B，項目 C，項目 D が定義されている．次にアプリケーション A のデザインシートを定義する．アプリケーション A では項目 D の設定値が共通デザインシートと異なる．したがって，項目 D を再度定義する．アプリケーション B では共通デザインシートには存在しない項目 E が必要となった．そこで新しく項目 E を定義するという考え方である．

汎化の概念を導入することで，共通デザインシートが 1 つ増えるが，アプリケーション A とアプリケーション B の LOI 値と CBD 値は大幅に改善される．特に，図 8 の汎化例では CBD 値が 0 となった．これは設定項目値のデザインシート間の関連がないことを示し，もし，IP アドレス等の設定項目値が変更になった場合でも，共通デザインシートの 1 カ所だけを変更すればよいことを意味する．汎化によって，1 つの設定項目値の変更を複数のデザインシートへ反映させる作業が不要となり，変更ミス等のヒューマンエラーを避けることができる．

ただし，1 つのデザインシートを共通部分と差分の 2 つのシートでの管理となる．2 つのシートを使い分けながら設計することは，現在の 1 つのシートでの設計より煩雑な作業となる．そこで，本改善策を実施するときは，デザインシート作成ツールを使用する．まず，共通デザインシートを読み込む．そこにはあらかじめ共通設定項目とその項目値が入力されている．設計時に，設定されている値を変更したり，新しく項目を追加したりした場合，共通シートとの差分が自動計算され，自動的に差分のみのデザインシートが生成される．このデザインシートを編集する際は，本ツールに読み込み編集して保存すると，新規の場合と同様に共通シートの差分のみがデザインシートとして保存される．今後，本ツールを開発し提供することで，容易に改善策の実施ができると考える．

6.2 コピーして作成されたデザインシートのメトリクス値の改善

上記の改善を 25 個のコピーして作成したデザインシートに適用した．25 個のデザインシートに共通デザインシートを設定し，25 個のデザインシートを共通デザインシートの差分だけを定義する方式に変更した．まず，共通デザインシートは Q1 のデザインシートをもとに作成した．Q1 のデザインシートの LOI 値 2,642 であり，Q1 のデザインシートを共通シートと仮定した．本来は最も共通項目が多いデザインシートをもとに共通デザインシートを作成すべきであるが，対象の 25 デザインシートはほぼ共通の項目であったので Q1 を共通デザインシートとした．

次に 25 個のデザインシートの差分を作成する．差分は，(1) 共通デザインシートに新しく項目を追加，(2) 共通デザインシートの設定された項目値の変更，(3) 共通デザインシートで空欄だった項目の値の追加の 3 種類に分類される．それぞれのデザインシートの 3 種類の差分の項目数とそれを合計した差分の項目数を汎化後 LOI 値として**表 2** に示す．

また，汎化後の新しいデザインシート間の CBD 値も同

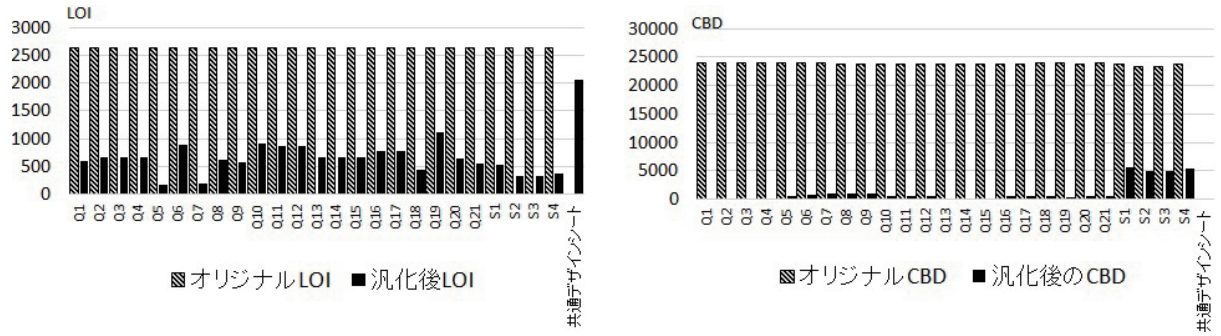


図 9 コピーして作成したデザインシートの汎化後の LOI 値と CBD 値
Fig. 9 LOI and CBD after generalization.

表 2 共通デザインシートとの差分の項目数
Table 2 The number of item of gaps in common design sheets.

デザインシート	(1) 追加項目数	(2) 変更項目数	(3) 空欄追加項目数	汎化後 LOI 値
Q1	0	0	0	0
Q2	5	206	451	662
Q3	5	206	450	661
Q4	5	206	451	662
Q5	22	133	27	182
Q6	17	810	59	886
Q7	17	129	51	197
Q8	6	283	337	626
Q9	7	271	307	585
Q10	6	644	264	914
Q11	6	642	224	872
Q12	7	383	490	880
Q13	9	611	50	670
Q14	10	611	50	671
Q15	12	611	50	673
Q16	14	667	107	788
Q17	14	667	107	788
Q18	22	346	82	450
Q19	17	1,084	6	1,107
Q20	13	392	236	641
Q21	15	269	274	558
S1	28	297	208	533
S2	24	79	225	328
S31	31	112	198	341
S4	33	132	212	377

時に計測した（図 9 参照）．LOI 値が 2,642 の共通デザインシートが 1 つ追加されたが，25 個のデザインシートは差分のみの定義となり，LOI 値は大きく減少した．さらに，設定項目の共通部分は共通デザインシートに記載されることになるので，デザインシート間に同じ項目値が出現することはない．しかし，差分の部分で定義される項目値に共通の値が出現したので，CBD 値はかなり減少したが，0 にはならなかった．いずれにしても，LOI 値，CBD 値ともに大きく改善することができた．

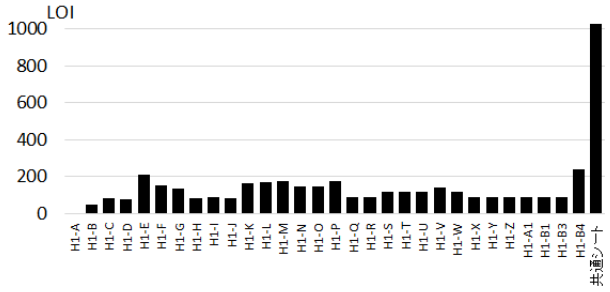


図 10 障害の発生したデザインシート H1 を分割した 30 個のデザインシートと共通シートの LOI 値
Fig. 10 LOI of the devided design sheets after generalization.

6.3 障害の発生した規模の大きいデザインシートのメトリクス値の改善

5.2.1 項で述べたように LOI 値が大きいデザインシート H1 で IP アドレス重複の問題が発生した．そこで，建屋ごとに設定されたデザインシートを分割し，30 個のエリアごとのデザインシートへ分割した．エリアとは VLAN で設定された範囲を示し，仮想的な LAN セグメントごとのデザインシートである．単純に分割しただけでは，デザインシートの総数が増えて，トータルの LOI 値は改善されず，反対に CBD 値は増加する．結果として，修正ミスにつながりやすい．そこで，30 個の新しいデザインシートに提案した汎化による LOI 値の改善を試みる．

デザインシートを VLAN エリアごとに分割した 30 個のデザインシート H1-A～H1-B4 と新しく生成された共通デザインシートの LOI 値を図 **10** に示す．6.1 節で説明した改善策により，1 つの共通デザインシートが新しく生成された．LOI 値は 1,024 である．基本的に H1-A の設定項目をベースに共通デザインシートを作成し，共通デザインシートとの差分を計測した．VLAN エリアには共通の設定項目が多く，項目の追加はごく少数であった．したがって，主な差分は設定項目値の書き換えであった．30 個に分割されたデザインシートの差分だけを記載すると，各デザインシートの LOI 値は最大でも 239 であった．30 個の分割されたデザインシートと新しく作成された共通シートのトータルの LOI 値は 3,499 であり，オリジナルデザインシート

の LOI 値 29,700 を大きく改善することができた．

また，1 つの大規模なデザインシートを 30 個のデザインシートと共通シートに分割したので，オリジナルデザインシートでは存在しなかった CBD 値が発生する．30 個のデザインシート間には共通の設定項目値が記載されるケースも発生した．つまり，共通デザインシートの差分で設定される値が共通値のときである．30 個の CBD 値を計測すると 1,049 となった．LOI 値が 9 割近く削減された利点を考慮すると，1,049 の CBD 値はけっして大きくないと考える．さらに，H1 デザインシートは規模が大きく，実際には内部に CBD に相当する関連が多く存在した．規模が大きいので一覧性が劣り，1 つの変更に対する他の箇所の変更箇所の特定が難しかった．デザインシートが分割されて，デザインシート間の関連が CBD 値で明確に計測されたことで，H1 デザインシートに埋もれていた内部の関連が明確になったことは意義があると考える．

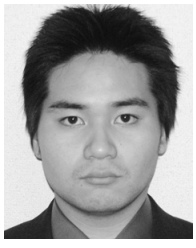
7. おわりに

インフラの品質を計測するメトリクス LOI，CBD を提案した．産業界のプロジェクトに適用した結果，コピーして作成したデザインシートの LOI がほぼ同値で CBD 値が極端に高いことが分かった．障害レポートとデザインシートの関係を調べた結果，LOI 値が高いデザインシートにはミスが潜む可能性があり，また，LOI 値が小さいにもかかわらず CBD 値が大きい場合，必要な項目の欠如の可能性も考えられた．したがって，設計時の質を維持するためのレビューを重点的に行う等の教訓を得られた．また，デザインシートの共通部分を抽出する汎化で共通デザインシートを設定し，各デザインシートを差分のみにすることで，LOI 値や CBD 値を改善する方法を示した．今後はデザインシートとコンフィグファイルとの関係を調査する予定である．

謝辞 本研究の一部は JSPS 科研費 JP26330093 の助成を受けた．

参考文献

[1] 松田晃一，紀夫鈴木三，大高 浩：情報システムの障害状況 2013 年前半データ，*SEC Journal*, Vol.9, No.3, pp.142–146 (2013).
[2] Hanakawa, N. and Obana, M.: System quality improvement including software and infrastructure based on software metrics approach, *Proc. Intl. Conf. 20th Asia-Pacific Software Engineering*, pp.9–16 (2013).
[3] 尾花将輝，花川典子，深海 悟：詳細設計書を用いたインフラストラクチャ構築の設定ミス予測のためのメトリクス NoCI の提案，第 20 回ソフトウェア工学の基礎ワークショップ（FOSE2013）予稿集，pp.77–82 (2013).
[4] 尾花将輝，花川典子：ソフトウェアメトリクスアプローチに基づくコンピュータシステムのインフラストラクチャ品質の検証，ソフトウェアエンジニアリングシンポジウム 2014（SES2014）予稿集，pp.137–142 (2013).
[5] DeMarco, T.: *Controlling Software Projects: Management, Measurement and Estimation*, Prentice Hall, Boston, USA (1983).
[6] 船越裕介，松川達哉，吉野秀明，後藤滋樹：通信ネットワークの保全度向上のための故障修理時間分布の特性分析，電子情報通信学会論文誌 B，Vol.J92-B, No.7, pp.1153–1163 (2009).
[7] 鳥山裕史，町澤朗彦，岩間 司：ハードウェア SNTP サーバの開発，電子情報通信学会論文誌 B，Vol.J89-B, No.10, pp.1867–1873 (2006).
[8] Clements, P., Kazman, R. and Klein., M.: *Evaluating a Software Architecture*, Addison Wesley, Boston, USA (2002).
[9] Hasegawa, M., Ishida, E. and Ogawa, H.: The establishment of the method for designing infrastructure to cope with changes quickly, *IBM Provision*, Vol.50, pp.68–75 (2006).
[10] Shi, H. and Zhan, Z.: An optimal infrastructure design method of cloud computing services from the BDIM perspective, *Proc. Intl. Conf. Computational Intelligence and Industrial Applications Asia-Pacific*, pp.393–396 (online), DOI: 10.1109/PACIIA.2009.5406408 (2009).
[11] Touzi, J., Benaben, F., Pingaudand, H. and Lorre, J.P.: A model-driven approach for collaborative service-oriented architecture design, *J. Production Economics*, Vol.121, No.1, pp.5–20 (2009).
[12] Pappas, V., Wessels, D., Massey, D., Songwu, L., Terzis, A. and Lixia, Z.: Impact of configuration errors on DNS robustness, *IEEE J. Selected Areas in Communications*, Vol.27, No.3, pp.275–290 (2009).
[13] Wool, A.: A quantitative study of firewall configuration errors, *IEEE Computer*, Vol.37, No.6, pp.62–67 (2004).
[14] 独立行政法人情報処理推進機構技術本部ソフトウェア・エンジニアリング・センター：非機能要求グレード利用ガイド，情報処理推進機構（オンライン），⟨https://www.ipa.go.jp/sec/softwareengineering/reports/20100416.html⟩（参照 2016-07-11）．
[15] 独立行政法人情報処理推進機構技術本部ソフトウェア・エンジニアリング・センター：非機能要求グレード活用シート，情報処理推進機構（オンライン），入手先 ⟨https://www.ipa.go.jp/sec/softwareengineering/reports/20130311.html⟩（参照 2016-07-11）．
[16] Hoyle, D.: *ISO 9000 Quality Systems Handbook - updated for the ISO 9001 standard (2008)*, Routledge, Kentucky, USA (2009).



尾花 将輝 （正会員）

1984 年生まれ．2007 年阪南大学経営情報学部卒業．2009 年同大学大学院修士課程修了．2013 年奈良先端科学技術大学院大学博士課程修了．博士（工学）．同年大阪工業大学助教．ソフトウェア工学の研究に従事．ソフトウェア科学会会員．

花川 典子

1961 年生まれ．1984 年大阪教育大学教育学部教育学科卒業．1997 年奈良先端科学技術大学院大学修士課程修了．2000 年同博士課程修了．博士（工学）．2002 年阪南大学講師．2003 年同大学准教授．2005 年同大教授．ソフトウェア工学プロセスの研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE 各会員．