Vol.2022-IOT-56 No.38
2022/3/8

マイクロサービス型システムの監視における ダッシュボードUI設計に起因する状況認識への影響

林 友佳[1,a)]　松原 克弥[1,b)]　鷲北 賢[2]　坪内 佑樹[2]

概要：システムの可用性や拡張性を高める設計手法のひとつとして，マイクロサービスアーキテクチャが注目されている．一方で，モノリシック型のシステムと比較すると，システムが大規模かつ複雑になり，運用や監視に対する負荷や難易度が高まるという課題がある．著者らは，マイクロサービス型システムの障害対応において，コンポーネント間の依存関係を把握することが最も重要であると考える．本研究では，コンポーネント間依存関係の可視化を中心としたシステム監視ダッシュボードを提案する．監視対象コンポーネント数の増加に対処するために，Situation Awareness と認知心理学の理論から導出した要件にもとづいて UI を設計した．本稿では，設計した監視ダッシュボード UI による状況認識の精度や認知負荷への影響を評価するために，実務経験を有する監視エンジニアを被験者とした，マイクロサービス型システムの障害対応に関するシミュレーションの実施計画について示す．

1. はじめに

マイクロサービスアーキテクチャを採用したシステムでは，各機能コンポーネントを複数のマシンに分散させ，ネットワークを介して接続することで 1 つのシステムを成り立たせている．この構築方法により，システムの負荷に応じてコンポーネントの数を増減させたりコンポーネントにエラーが発生した際に交換したりすることが容易になり，システムの可用性や拡張性を向上させることができる．一方で，モノリシック型のシステムと比較して，システムが大規模化・複雑化することで，運用や監視に対する負荷や難易度が増す課題が存在する．

このマイクロサービス型システムの監視における最大の課題のひとつは，コンポーネントの数や稼働状況が頻繁に変化するために，システム内の状態変化を把握することが難しいことである．システムがアクセス負荷上昇やコンポーネント異常終了などに対処することで，コンポーネントの数，コンポーネントが稼働するマシンノード，コンポーネント間の連携状況が変化する．したがって，個々のコンポーネントのメトリックを監視しているだけでは，これらの状況変化の発生やその理由を特定することができず，結果として，システム全体の状態を把握できないことにつながる．

また，マイクロサービス型システムでは，監視すべきコンポーネントの数が多いことから，障害原因の特定が難しくなる課題も存在する．マイクロサービス型システムのような分散システムでは，コンポーネント間の連携にともなう暗黙的な依存関係の発生により，障害がコンポーネント間を伝播する [1]．前述の状態変化にともなってコンポーネント間の依存関係も変化することから，障害原因の特定のために分析すべきコンポーネントの範囲を特定することも困難となる．

著者らは，マイクロサービス型システムの状態監視や障害対応において，コンポーネント間の依存関係を把握することが最も重要であると考える．そこで，本研究では，コンポーネントの依存関係を中心としたシステム監視ダッシュボードを提案する．コンポーネントとその依存関係の変化を可視化することで，システム全体の状態把握と変化の認知を支援する．さらに，可視化したコンポーネント間の依存関係に合わせて，各コンポーネントのメトリックを表示し比較する機能を実現することで，障害の伝播をトレースすることを支援できる．

依存関係を中心とした監視ダッシュボード User Interface（以降，UI）では，コンポーネントの数に応じて表示する情報量や操作の対象が指数関数的に増加する．そのため，監視エンジニアが処理すべき情報量の増加が，システム状態に関する認知へ影響を与える可能性がある．本稿では，設計した監視ダッシュボード UI による状況認識の精度や

[1] 公立はこだて未来大学
Future University Hakodate, Hakodate, Hokkaido, Japan
[2] さくらインターネット株式会社
SAKURA internet Inc.
[a)] g2120042@fun.ac.jp
[b)] matsu@fun.ac.jp

認知負荷への影響を評価するために，実務経験を有する監視エンジニアを被験者とした，マイクロサービス型システムの障害対応に関するシミュレーションの実施計画について示す．

本稿を次のように構成する．2 章では，マイクロサービス型システムの監視における課題について述べる．3 章では，人の状況認識や意思決定のプロセスに着目した Situation Awareness（以降，SA）や認知心理学などの理論に基づいて，ダッシュボードの UI 設計のための要件を導出する．4 章では，依存関係を中心としたシステム監視ダッシュボードの UI を提案する．5 章では，既存の監視ツールと提案手法の監視ダッシュボードを用いたアンケート調査によって各要件が認知へ与える影響について予備評価の結果を示す．6 章では，実務経験を有する監視エンジニアを被験者とした，マイクロサービス型システムの障害対応に関するシミュレーションの実施計画について述べる．7 章では，本稿をまとめ，今後の展望を述べる．

2. マイクロサービス型システムの監視における課題

マイクロサービス型システムの監視における課題は 2 つある．第 1 の課題は，マイクロサービス型システムでは，障害発生時に監視エンジニアにかかる認知的な負荷が高くなる傾向があることである．マイクロサービス型システムなどの分散システムでは，コンポーネントの依存関係が影響し，複数の要因が組み合わさることで障害が発生する傾向がある [2]．さらに，依存関係があるコンポーネントに障害が伝播することがあり，監視ツールはそれぞれのコンポーネントごとにアラートを発するため，システム全体では大量のアラートが発生することになる．このような大量のアラートは，システムを監視するエンジニアに疲労を与えたり思考を阻害したりするため，障害の根本的な原因を把握することが困難になり，障害への対応に時間を要してしまう原因となる．言い換えれば，短期間に確認すべき情報が大量に発生したり重要度がわからない情報が複数存在したりすると，情報の認識や取捨選択が監視エンジニアに対する認知的な負荷になり，障害への迅速な対応が難しくなるということである．そのため，監視エンジニアにかかる認知的な負荷を軽減するための工夫が必要である．

第 2 の課題は，障害の影響が広がる範囲や，今後システムにどのような変化が起こるかといった，予測や見積もりが難しいことである．マイクロサービス型システムは依存関係が頻繁に変化するため，監視エンジニアがシステム全体の依存関係を常に把握することは難しい [1], [3]．その上，個々のコンポーネントに関して，CPU 使用率やメモリ使用量，リクエスト数，レイテンシなど，監視メトリクスの値の変化も発生するため，システム全体として変動する要素が多い．結果として，システムの状態の把握が難しくなり，障害による影響範囲や影響の内容の予測に必要な情報を見逃してしまうことがある．そのため，障害による影響を予測するために必要な情報を適切に提供すべきである．

3. 監視ダッシュボードの UI の設計要件

前章で述べた課題を踏まえて，監視ダッシュボードの UI 設計では，監視エンジニアに対する認知的な負荷の軽減や，今後の予測を支援するような情報提示が必要であると考える．さらに，障害の原因追求や今後の予測のための情報収集を迅速に行えるよう，ツールの操作性についても工夫が必要であると考える．これらを達成するために，本章では，監視エンジニアに対する認知的な負荷の軽減や，今後の予測を支援する情報提示が可能なダッシュボードを設計するための要件を検討する．ここでは，今後の予測や，予測をもとにした意思決定のために，複数の情報源から複雑な状況を短時間で認識する必要があることに着目し，そのような状況認識のプロセスを扱う SA の理論を活用する．また，認知的な負荷の軽減や円滑な情報収集の支援を実現するには，人間が情報を認知する方法を理解し，その方法に合わせた情報の可視化手法を検討する必要があるため，認知心理学に関する理論も活用する．以降では，マイクロサービス型システムを監視するダッシュボードの UI 設計に適用すべき SA や認知心理学に関する理論や法則について整理し，要件をまとめる．

3.1 SA に関する設計法則

SA とは，一定の時間と空間の中で環境にある要素を認識し，その意味を理解し，近い将来の状況を予測するプロセスのことである [4]．Endsley[4] は，認識，理解，予測の段階をそれぞれレベル 1〜3 に分けてモデル化して扱っている．レベル 1〜3 のそれぞれの段階を確実に行うことが，次に起こすべき行動や意思を決定するために重要である．言い換えると，いずれかのレベルに関する能力や必要な情報が欠けている場合は，正しい状況認識ができず，正しい意思決定ができなくなる可能性がある．SA の概念は，航空機や航空交通管制，製造システムや原子力発電所などの大規模システムの操作，医療などの領域で研究され，活用されてきた．システム監視と障害対応に，SA から意思決定までのプロセスを適用すると，以下のようになる．

レベル 1 SA　一定の時間内で，アラートや監視ダッシュボードの表示から，システムのどの部分にどのような変化が起きているかを認識する

レベル 2 SA　認識した情報を集め，因果関係を踏まえて整理することで，システム全体の現状を理解する

レベル 3 SA　対応をしないとシステムが今後どうなるか，また，どのような対応をするとどのような結果になるかについて，現状の理解に基づき，予測する

意思決定　予測を基に，実際に行う対応の内容を決定する

SA の観点を取り入れたシステム設計を SAOD（Situation Awareness Oriented Design）という．Endsley らは，システムのユーザの SA を支援し強化するために重要な，SAOD に関わる 50 の設計法則を提案している [5]．50 の設計法則のうち，チームで共有するシステムに関する法則，自動化に関する法則，アラームなど視覚的な UI 以外に関する法則，不確実な情報が収集された際に関する法則を除いた，16 個の法則がシステム監視のダッシュボードの設計に適用可能と考える．以下に，適用すべき設計法則の一部を抜粋する．

- Organize information around goals
  情報は，情報を作成したセンサーやシステムに基づいて表示される技術志向の方法で提示するのではなく，ユーザの主要な目的に基づいて編成する．
- Present Level 2 information directly - support comprehension
  ユーザの注意やワーキングメモリは限られているため，レベル 2 SA の要件に基づいて予め処理，統合した情報を提供する．
- Provide assistance for Level 3 SA projections
  過去の値の変化や，情報から予測できる内容を表示することで，ユーザが今後のシステムの状態や周囲の環境を予測できるように支援する．
- Use information filtering carefully
  SA に不必要な情報や，未処理のデータの削減は有益であるが，全体像の把握に必要な情報や，予測に必要な情報，必要性の有無に個人差があるような情報の削減は注意する．
- Group information based on Level 2 and 3 SA requirements and goals
  ユーザが目的達成に必要な SA を導出する際に，複数のシステムから横断的に探す必要がないように，同じ目的をサポートするために使われる情報は同じ表示領域に集中させる．
- Minimize task complexity
  ユーザの認知負荷やミスを軽減するため，必要なタスクを実行するために必要なアクションの数とアクションの複雑さを最小限に抑える．

3.2 認知心理学に関する設計法則

ダッシュボードの設計を検討するためには，SA に関する法則に加えて，提供すべき情報を可視化する具体的な方法についても検討する必要がある．そこで，認知心理学的観点からダッシュボード上での情報の可視化手法や UI の設計について検討し，認知的な負荷を軽減することを目指す．本論文では，Few[6] が述べる，ダッシュボードの設計で一般的な 13 の間違いや，ダッシュボードの設計をする際に気をつけるべき点，活用すべきデザインの法則を参考に可視化手法や UI を検討する．

例えば，監視ツールのダッシュボードでは，エラーが発生しているコンポーネントや定常状態より負荷が高いことを示すデータなどを強調することで，エンジニアが迅速に認知できるようにすべき場面がある．このような場面において，重要な要素を強調しなかったり，多すぎる要素を強調させるような可視化は，ユーザを混乱させて，認知的な負荷をかけてしまう．強調したい要素を絞り，その箇所のみ鮮やかな色を使用したり，色，位置，形，動きを他の要素と異なるよう設定したりすることで，適切な強調が可能である．一方で，一貫性や統一感を持たせるべき要素については，色，位置，形，動きを統一することで，認知的な負荷を軽減することができる．

他にも，監視ツールのダッシュボードでは，複数のコンポーネントに対して，関連する監視メトリクスの確認や比較などの操作を行うことがある．その際に，ユーザが一目で見ることができる範囲内に重要な情報がないことは，ユーザに様々な箇所を探させる手間や負荷をかけてしまう可視化手法である．ユーザに負荷をかけずに重要度の高い情報を認識させるためには，ユーザの視線が向きやすいダッシュボード上の左上や中央に重要な情報を表示することが望ましい．監視メトリクスの確認などの操作に関しては，ユーザがデータに対して直接インタラクションすることで追加データにアクセスできるように設計することで，前節の法則にあったように必要なアクションの数を最小限にすることができ，認知負荷や操作ミスを軽減することができる．監視メトリクスの比較に関しては，グラフを結合する，比較したい要素同士を近くに配置する，比較したい要素同士に共通の色を用いてリンクさせる，比較用の値を表示する，などの工夫を行うことで比較作業を支援できる．このように比較を支援することで，比較のために様々な情報を探す際にかかる負荷を軽減できるだけでなく，過去の情報の比較から今後の予測をする際にも役立つ．

また，認知的な負荷を軽減した上で多くのデータを可視化する際には，視覚におけるゲシュタルトの法則を活用したグループ化が有効な手段である．視覚におけるゲシュタルトの法則とは，オブジェクトを一緒にグループ化する傾向がある視覚的特性を下記のようにまとめたものである．

I. The Principle of Proximity
   近くにあるオブジェクト同士を同じグループとして扱う傾向がある．
II. The Principle of Similarity
   似た色，大きさ，形，方向のオブジェクトを同じグループとして扱う傾向がある．
III. The Principle of Enclosure
   線で囲む，背景に色を付けるなどによって，境界があるとその境界ごとにグループとして扱う傾向がある．
IV. The Principle of Closure

完全に閉じられていない形でも，閉じられた規則的な形として認識する傾向がある．

V. The Principle of Continuity
インデントが揃っているオブジェクトなど，連続性があるように見えるオブジェクトを同じグループとして扱う傾向がある

VI. The Principle of Connection
線などで繋がれているオブジェクトを同じグループとして扱う傾向がある．I，II より強く働き，III よりは弱い働きを持つ．

3.3 設計法則から導出した要件

以上のような法則を基に，認知的な負荷を下げるためのダッシュボードの設計要件を以下の通り定義する．

C1 情報はグループ化や適切なフィルタリングなどがされた状態で表示されている
認知的負荷を軽減した上で，頻繁に変化するシステムの状態や依存関係を把握，理解できるようにするため

C2 一貫性を持たせたい要素同士は，同じ色，形，動きなどで構成されている
ユーザの認識に不要なノイズを与えないようにすることで、認知的な負荷を軽減するため

C3 強調したい要素は，他の要素とは異なる色，形，動きなどをつけることで他の要素と区別されている
ユーザに認知的な負荷をかけずに，重要な情報に気づかせるため

C4 ダッシュボード上の強調される位置である左上か中央に，重要度の高い情報が表示されている
ユーザに認知的な負荷をかけないようにした上で，重要な情報を見落とさないようにするため

また，今後の見積もりを支援するためのダッシュボードの設計要件を以下の通り定義する．

C5 特定の瞬間のデータだけではなく，前後のデータも表示されている
頻繁に変化するシステムの状態の理解と今後の予測に役立てるため

C6 意味のある比較を支援している
あるコンポーネントの過去の状態や，依存関係がある別のコンポーネントと比較できるようにし，今後の予測に役立てるため

さらに，ツールを迅速に操作できるよう支援するためのダッシュボードの設計要件を以下の通りまとめた．

C7 システムの状態確認や依存関係の把握のために必要な操作は複雑すぎない
迅速に必要な情報を入手できるようにするため

C8 ボタンなどではなく，表示されているデータに何かしら操作をすることで，データの詳細などが表示できるようになっている

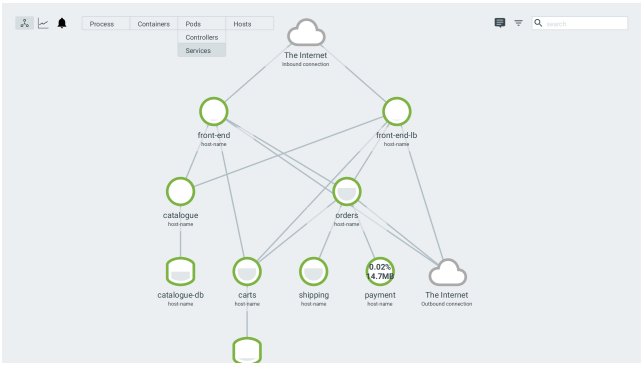


図 1 提案したダッシュボードの依存関係可視化画面

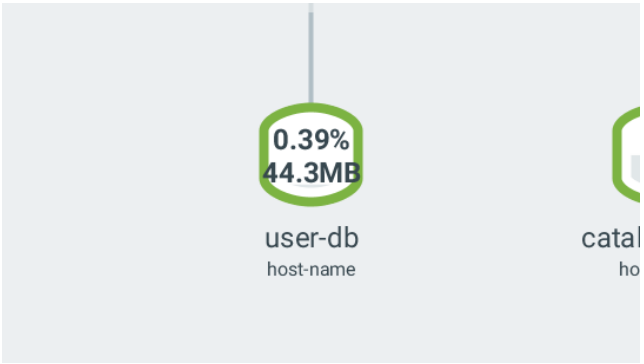


図 2 マウスホバー時にコンポーネントを表す図形上に CPU 使用率とメモリ使用量が表示される様子

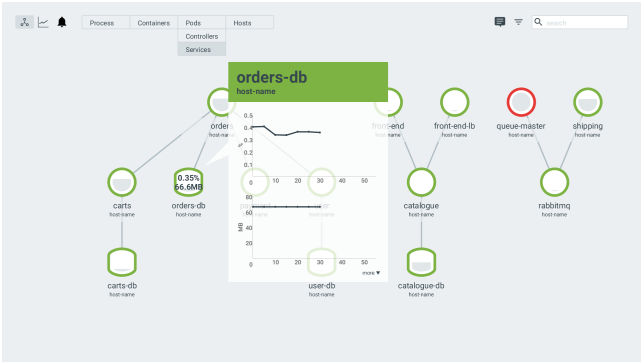


図 3 コンポーネントを表す図形をクリックした際に表示される，CPU 使用率とメモリ使用量のグラフ

ユーザが直感的に，ミスなく追加データにアクセスできるようにするため

4. ダッシュボードデザインの提案

本章では，前章で述べた要件に基づき，マイクロサービス型システムの状態可視化に適した可視化手法を提案する．本稿で提案するダッシュボードの UI は，コンポーネントの地理的・時間的な状態変化を同時に可視化することと，視覚的示唆による強調表現と俯瞰的可視化の両立を行うという特徴がある．

依存関係の変化などの地理的な状態変化は，各コンポーネントを円などの図形として描画し，ネットワーク通信関

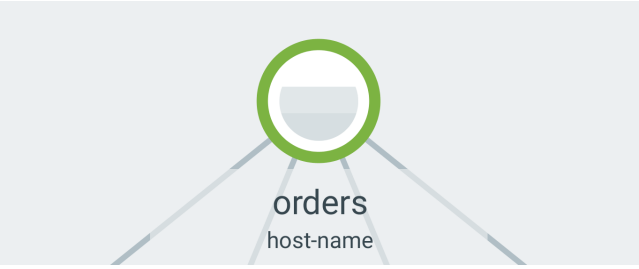


図 4　CPU 使用率とメモリ使用量の値に応じて，コンポーネントを表す図形の内部が塗りつぶされている様子

係にあるコンポーネント間に線を描画することで表す（図 1）．依存関係のあるコンポーネントを線でつなぐことは，ゲシュタルトの法則のうち VI の接続の原則を使っているため，C1 を満たすことになる．コンポーネントやそれに伴う依存関係が，特別な操作なしに一覧できるような画面になっているため，C4, C7 を満たす．それぞれのコンポーネントを表す図形にマウスカーソルを重ねると，そのコンポーネントの CPU 使用率とメモリ使用量の数値が表示される（図 2）．直感的かつ単純な操作で状態把握が可能なため，C6, C7 を満たしているといえる．

コンポーネントを表す図形をクリックした際には，CPU 使用率やメモリ使用量などの時間ごとの値を表す折れ線グラフを描画することで，時間的な状態変化を表す（図 3）．こちらについても，対象のコンポーネントに対する直接的かつ単純な操作で情報を得られるため，C7, C8 を満たす．さらに，時系列データを表示できるため，C5 を満たす．複数のコンポーネントの折れ線グラフを表示した後ドラッグをすることで，グラフを重ねることができ，1 つのグラフで複数コンポーネントの時系列データを確認することができる．これにより，1 つのグラフで複数コンポーネントの値の比較が可能になるため，C6 を満たす．また，特定のコンポーネントを指定して，表示時間を巻き戻すよう指定することで，そのコンポーネントに関する過去の依存関係を描画する．これもまた，特定の瞬間だけでなく前後の時間の情報にアクセスできることになるため，C5 を満たす．これらの機能によって，コンポーネント間の依存関係やその変化，それぞれの監視メトリクスの変化を迅速に認識できるようにしている．

視覚的示唆による強調表現は，コンポーネントを表す図形の調整や，色使いによって行っている．コンポーネントを表す図形は，コンポーネントの種類別に変更している．例えば，図 1 では，通常のサービスの処理に関わるようなコンポーネントとデータベース，インターネットのそれぞれを，異なる形を用いて表していることがわかる．言い換えると，同じコンポーネントについては一貫性を持つように形を統一しているため，C2 を満たす．また，同じコンポーネントをゲシュタルトの法則のうち類似性の法則を用いてグループ化しているという側面もあるため，C1 も満たす．コンポーネントを表す図形の描画位置は，基本的には，データのやりとりの流れに沿ってコンポーネントを配置している．その他，ユーザの要求に合わせて，ユーザが指定した特定のコンポーネントを上部に配置したり，フィルタリングを行い，一部のコンポーネントのみを表示したりすることも可能である．フィルタリングはその名の通り C1 を満たすことにつながり，ユーザが指定したコンポーネントを上部に配置することは C4 を満たすことにつながる．

色使いについては，コンポーネントを表す図形の枠線の色と，図形の内部の色にそれぞれ意味を持たせている．枠線の色は，CPU 使用率やメモリ使用量の値が大きい場合は赤色にし，値が中くらいの場合は黄色，小さい場合は緑色とするといったように区別している．例えば，図 1 では，queue-master のみコンポーネントを表す図形の枠線が赤く表示されているため，負荷の高いコンポーネントであることがわかる．コンポーネント内部は白とグレーで塗りつぶされ，グレーの面積によって，CPU 使用率とメモリ使用率の値が示唆される（図 4）．監視エンジニアにとって重要な要素を色によって強調しているため，C3 を満たす．

5.　UI 設計要件に関する予備評価

前章で導出した要件の妥当性に関して，各要件が監視エンジニアの受け取る印象や状況の認知に与える影響を，監視ツールの画面を用いたアンケートを実施することで評価した．アンケートの対象はインフラ構築やシステム監視に携わった経験のある人で，学生 2 名と企業のエンジニア 6 名の計 8 名から回答を得た．

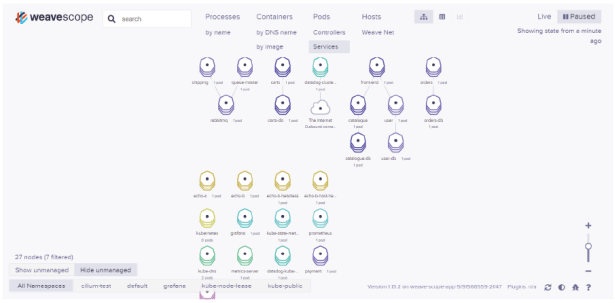
5.1　アンケートの概要

評価では，特定のマイクロサービス型システムの監視画面として，要件を満たした画面と満たさない画面を提示し，それぞれに対して印象や認知を問うアンケートを実施した．提示する画面には，既存の監視ツールである Weave Scope[7] と，提案手法の監視ダッシュボードを使用している．また，今回の評価では，静止画でも検証可能な要件のみを対象としており，該当する要件は C1, C2, C3, C4 の 4 つである．

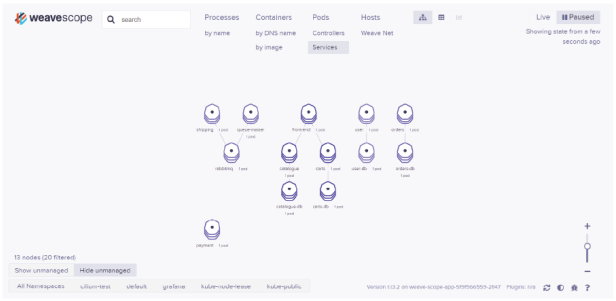
アンケート内で提示した画面を図 5 に示す．画面 1 は，C1 から C4 の全ての要件を満たすように作成した提案手法の画面である．C1 を満たすために，依存関係のあるコンポーネント同士を線でつなぐ，同じ役割のコンポーネント同士を同じ図形で可視化する，監視対象として必要なコンポーネントのみ表示するなどといった手法によりグループ化やフィルタリングを行っている．C2 を満たすために，同じ役割のコンポーネント同士を同じ図形で可視化し，コンポーネントの CPU 使用率やメモリ使用量の値に応じて図形の枠線の色を設定することで，一貫性を持たせている．C3 を満たすために，コンポーネントの CPU 使用率やメモ

画面1

画面2



画面3



画面4

図 5　アンケートで提示した画面の画像

リ使用量の値が高く障害の原因となりうるコンポーネントは，図形の枠線の色を赤色にすることで強調する一方で，他のコンポーネントはその赤色ほどは目立たない色を設定している．C4 を満たすために，中央の広い領域にコンポーネントの依存関係を可視化するとともに，左上から順に依存関係を多く持つコンポーネントを配置している．画面 2 は，提案手法をもとにした画面であるが，図形の色や形をランダムに決定しているという特徴がある．監視対象として必要なコンポーネントのみグループ化して中央に可視化していることから，C1 と C4 は満たす一方で，ランダム性によって C2 や C3 は満たさない．画面 3 は，Weave Scope の画面であり，図形の形や色に一貫性はある一方で，表示されているコンポーネント数に対して形や色の種類は少なく，C1, C2 を満たす度合いが提案手法よりも低い．また，意図的に表示するコンポーネント数を増やしており，システムの状況に応じた強調などはなく，重要でないコンポーネントが画面左上や中央に表示されているという特徴があるため，C3 や C4 に反する箇所があるといえる．画面 4 は，Weave Scope の画面であり，先程の画面 3 よりも表示するコンポーネント数が少なく，画面 1 や画面 2 と同じシステムのコンポーネントを可視化している．そのため，C1, C2, C4 を満たしているが，画面 3 と同様システムの状況に応じた強調などはないため，C3 には反する画面となっている．

印象についての質問項目は，「A. 乱雑な印象がある」「B. すっきりしていて見やすい」「C. 一貫性があるように感じる」「D. 図形の枠の色の意味がわかる」という質問に対し，それぞれ「とてもそう思う」「そう思う」「どちらとも言えない」「そう思わない」「全くそう思わない」の 5 段階で回答する形式であった．認知についての質問項目は，画面上のコンポーネント全てに番号を振った画像を提示し，その画面上で 1 番負荷が高いコンポーネントはどれだと思うかという質問に対し，振られている番号，もしくは「わからない」のいずれかを回答するとともに，理由を自由記述で任意に回答する形式であった．

5.2　結果と考察

印象に関するアンケート項目の結果を図 6 に示す．

C1 については，満たしていない画面 3 について「1 番負荷が高いコンポーネントはどれか」に対してわからないという回答が 75%であった．また，「A. 乱雑な印象がある」に対してとてもそう思うとそう思うの合計割合が高く，「B. すっきりしていて見やすい」に対して全くそう思わないとそう思わないの合計割合が高い．これらの結果から，C1 を満たさないことは，システムの状況認識のしやすさに悪影響を与えることが示唆される．

C2 については，満たしていない画面 2 について，「1 番負荷が高いコンポーネントはどれか」の正解率が 0%であった．また，「A. 乱雑な印象がある」に対してとてもそう思うの割合が他の画面よりも高く，「C. 一貫性があるように

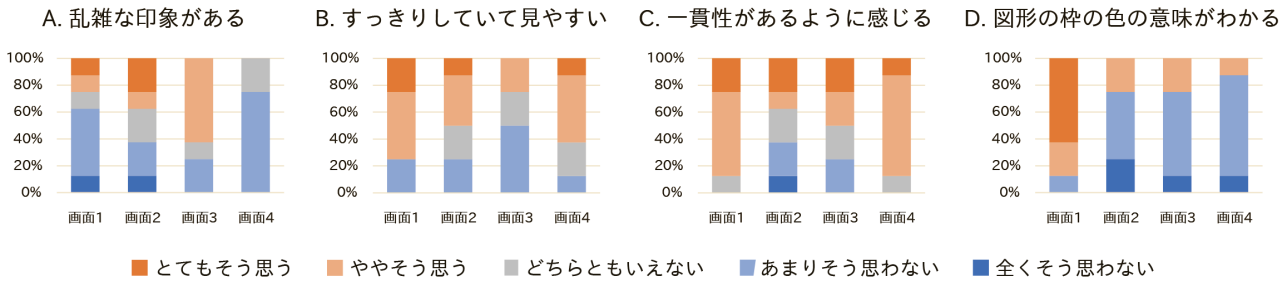


図 6　アンケート結果

感じる」に対する回答にばらつきがあるという特徴がある．これらの結果から，C2 を満たさないことで，システムの状況認識を誤認させる可能性があると考える．

C3 については，満たしている画面 1 について「1 番負荷が高いコンポーネントはどれか」に対して正解率が 100%であった．また，「D. 図形の枠の色の意味がわかる」に対してとてもそう思うの割合が高いことから，適切な強調ができていると考えられる．これらの結果から，C3 を満たすことで，正しくシステムの状況を認識できることに繋がると考える．

C4 については，Weave Scope の画面の「最初に目に入った/視線が向いた箇所はどこか」について，50%の人が画面左上に配置されていたロゴの箇所を選択した．その理由として，「初めて見る画面だったので、全体をぼやっと見ていた後、目についたのが左上ロゴだった」といった回答があり，重要な情報を載せるべき画面左上にロゴがあることでそちらに視線が向いたと考える．さらに，「他の箇所は色がついていないのに対してロゴは色がついている。」といった回答もあり，C3 のような強調効果がロゴにのみ働いてしまったことも影響していると示唆された．これらの回答から，C4 を満たさないことで，状況認識の優先度に悪影響を与えることが示唆される．

以上の結果と，全ての要件を満たす画面 1 では，「1 番負荷が高いコンポーネントはどれか」の正解率が 100%であったことから，要件を満たすことでシステムの全体や障害の原因となりうるコンポーネントの認識がしやすくなると言える．そのため，今回評価した要件 C1 から C4 は妥当であったと考える．

6. 障害対応シミュレーション

本章では，今後実施予定であるマイクロサービス型システムの障害対応をシミュレーションする評価実験について述べる．既存の監視ツールと提案手法を用いて障害対応をシミュレーションすることで，原因特定にかかった時間や調査の順序を比較し，マイクロサービス型システムの監視に対する提案手法の有効性を示す．

6.1 実験環境

テストベッド　本実験では，Kubernetes クラスタ上に，マイクロサービス型システムのデモンストレーションアプリケーションである Sock Shop を構築する．Kubernetes クラスタは 1 台の Master ノードと 3 台の Worker ノードで構成される．Master ノードには既存の監視ツールや負荷生成ツールを配置し，Worker ノードには Sock Shop のコンポーネントや監視メトリクス収集ツールを配置する．

アプリケーション　Sock Shop は靴下を販売する e コマースサイトを，マイクロサービスアーキテクチャを用いて構築したアプリケーションである．Sock Shop は，carts，catalogue，front-end，orders，payment，shipping，user という 7 つのマイクロサービスからなる．各マイクロサービスのコンテナは 1 つずつ生成されるように設定する．

監視メトリクスの収集　既存の監視ツールと提案手法で可視化する，監視メトリクスのデータ収集には Prometheus を使用する．Prometheus は，監視メトリクスを時系列データとして収集し保存する監視ツール兼時系列データベースである．Prometheus により，各コンテナの CPU，メモリ，ネットワーク，ディスクといったリソースの使用量や，各マイクロサービスの時間あたりの処理リクエスト数や平均応答時間などといった監視メトリクスを収集する．監視メトリクスの収集は 1 分毎に実施するよう設定する．

依存関係情報の収集　提案手法で可視化する，Sock Shop のマイクロサービス間に発生する依存関係情報は，Weave Scope の API を経由して収集する．Weave Scope はリアルタイムの情報のみを提供しており，依存関係情報や監視メトリクスをデータベースに保存する機能がないため，API のレスポンスを 2 秒毎に JSON 形式で保存して使用する．また，API のレスポンス内容には，本実験で使用しないデータも大量に含まれているため，扱いを容易にするために，タイムスタンプと依存関係のあるコンテナ名のみを抜き出すスクリプトを Python で記述し，1 時間分の依存関係情報を 1 つの CSV ファイルとしてまとめて使用する．

負荷生成　Sock Shop に対して擬似的な負荷を生成するために，Locust を使用する．Locust は，Python のコードでユーザの振る舞いをシナリオに記述することで，記述内

容に沿って自動で Web サイトにアクセスし，擬似的な負荷を生成するツールである．本実験では，Sock Shop の配布元が Sock Shop 用のシナリオとして配布しているコードを利用することで，実際にユーザが Sock Shop 上で商品を選んだり注文したりするような負荷を生成する．

障害生成　Sock Shop に対して擬似的な障害を発生させるために，stress-ng や tc（Traffic Control）を使用する．関連研究 [1], [8] の実験を参考に，stress-ng を利用して CPU やメモリの使用率を高めるような負荷をかけたり，tc を利用してパケットロスを発生させたりすることで，擬似的な障害とする．

既存の監視ツール　提案手法との比較対象として，既存の監視ツールである Zabbix と Grafana を使用する．

6.2 実験手順

実験の被験者として，実務経験を有する監視エンジニアに対して，下記の手順でオンライン実験を実施する．

手順 1　事前アンケートに回答してもらう
手順 2　実験時の教示事項などを伝える
手順 3　1 つ目の監視ツールを操作してもらう
手順 4　1 つ目の監視ツールに関するアンケートに回答してもらう
手順 5　3〜4 を計 3 回（それぞれ Zabbix，Grafana，提案手法に対して）実施する
手順 6　事後アンケートに回答してもらう

すべてのアンケートは Google フォームを用いて Web ブラウザ上で回答してもらい，手順 2〜6 は Zoom を用いて被験者とオンラインで会話しながら進める．手順 1 では，被験者の監視経験年数や使用したことのある監視ツール，実験当日に使用するディスプレイ枚数などを尋ねる．被験者が監視ツールに慣れている度合いや被験者側の実験参加環境を統一できない都合上，事前にこれらの項目を聞き取っておくことで，考察の際に活用することを想定している．実験時の教示事項では，実験で使用する監視ツールについての説明をするとともに，被験者の視線の流れや思考を把握するために，監視ツールの操作画面を共有しつつマウスカーソルを被験者自身の視線の移動に合わせて移動させるほか，その時に考えていることをできる限り口頭で話しながら操作するよう指示する．手順 3 の開始時に，前節で述べた障害を発生させる．その障害の根本原因が特定できた時点で被験者に申告してもらうことで，障害発生から障害の根本原因特定にかかる時間を計測する．そして，特定した障害の根本原因や，ツールを使用した際に感じたことについて手順 4 のアンケートで回答してもらう．手順 4 のアンケートの内容には，SUS（System Usability Scale）[9] を含めることで，3 種類の監視ツールのユーザビリティを計測する．手順 6 の事後アンケートでは，3 種類の監視ツールを比較して，障害の根本原因を探しやすいと感じた順に順位をつけてもらう．

7. おわりに

本稿では，マイクロサービス型システムを対象とした監視ダッシュボード UI を提案し，UI 設計が状況認識の精度や認知負荷に与える影響を評価するためのシミュレーションの実施計画について述べた．提案までの過程として，SA や認知心理学の理論により，マイクロサービス型システムの監視ダッシュボードの UI 設計に関わる 8 個の要件を導いた．また，予備評価では，導出した要件のうち，一部の要件についてアンケート調査を行った結果，要件の設定が妥当であることが示す結果が得られた．

今後は，障害対応シミュレーションを実施し，提案手法と既存の監視ツールを比較することで，提案手法の利点や有効性を示す．さらに，既存のオープンソース監視ツールのダッシュボードを本研究の要件に基づいた可視化手法に差し替えたり，既存のオープンソース監視ツールと本研究で設計するダッシュボードを連携させたりすることで，監視エンジニアがより迅速かつ容易に障害に対処できるよう支援したい．

参考文献

[1] Tsubouchi, Y., Furukawa, M. and Matsumoto, R.: Transtracer: Socket-Based Tracing of Network Dependencies Among Processes in Distributed Applications, *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1206–1211 (2020).
[2] Yuan, D., Luo, Y., Zhuang, X., Rodrigues, G. R., Zhao, X., Zhang, Y., Jain, P. U. and Stumm, M.: Simple testing can prevent most critical failures: An analysis of production failures in distributed data-intensive systems, *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 249–265 (2014).
[3] Neves, F., Vilaça, R. and Pereira, J.: Black-Box Inter-Application Traffic Monitoring for Adaptive Container Placement, *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, Association for Computing Machinery, p. 259–266 (2020).
[4] Endsley, M.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal, *Human Factors: The Journal of the Human Factors and Ergonomics Society*, Vol. 37, pp. 32–64 (1995).
[5] Endsley, M., Bolté, B. and G., J. D.: *Designing for Situation Awareness: An Approach to User-Centered Design*, Taylor & Francis (2003).
[6] Few, S.: *INFORMATION DASHBOARD DESIGN*, O'Reilly Media, Inc. (2006).
[7] Weaveworks, Inc: Weave Scope: Automatically Detect and Process Containers And Hosts (online), available from 〈`https://www.weave.works/oss/scope/`〉 (accessed 2022-02-01).
[8] 近藤玲子，麻岡正洋，渡辺幸洋：マイクロサービス運用支援のためのコンテナからインフラまでの構成差分検出技術の開発，研究報告インターネットと運用技術（IOT），Vol. 2020-IOT-49, pp. 1–6 (2020).
[9] Brooke, J.: *"SUS-A quick and dirty usability scale." Usability evaluation in industry*, CRC Press (1996).