第 6 分科会　B グループ

データベースを共有し合うシステム間での変更の影響を効果的に検知する方法

Effective way to detect impacts of changes among database sharing systems

主査　　：飯泉　紀子（株式会社日立ハイテクノロジーズ）
副主査　　：足立　久美（株式会社デンソー）
アドバイザー：清水　吉男（株式会社システムクリエイツ）
リーダー　　：林　慎一郎（東京海上日動システムズ株式会社）
研究員　　：藤井　伸之（テックスエンジソリューションズ株式会社）

研究概要
エンタープライズ系システムでは，顧客に付加価値の高いサービスをスピーディに提供するために，複数のシステムがデータベースを共有するようになった．これに伴い，派生開発時に影響範囲を特定することが難しくなり，データベース関連のシステム間不具合が発生するようになった．過去に発生した不具合事例を分析したところ，共通する発生要因と，設計担当者が不具合の可能性を見過ごしやすい設計変更パターンが存在した．我々はこの問題を解決するために，変更要求仕様と関連システム間のトレーサビリティマトリクスにデータベース変更時の必須チェック機能を組み込むことで，データベース変更時の影響の特定漏れを低減する方法を考えた．この方法を過去の不具合事例に適用してシミュレーションした結果，設計工程で変更漏れの検知に有効であることを確認した．

Abstract
In enterprise systems, database sharing systems has become the mainstream for providing high value-added services to customers speedily, As a result, it became difficult to identify the effect on the database sharing systems and problems for the database sharing systems are adapted to produce. Our team analyzed the problems occurred in the past case. As a result, there is common generation factor and design change patterns easy to personnel overlook the possibility for problems. To solve this problem, our team considered a method of reducing the specific leakage effects of the database changes for using traceability matrix between related systems and change required specifications with the check function for specific one database sharing system. And we have simulated by applying this method to the past problems cases. As a result, our team confirmed to be effective in the detection for change leakage in the design process.

1.　はじめに

エンタープライズ系システムではサービスや部門単位に個別システムを開発・運用しており，必要に応じて関連するシステム間でデータ連携してきた．しかし，サービス間のデータをシームレスに使用して付加価値の高いサービスをスピーディに提供するために，複数のシステムがデータベース（以降，DB と記載する）を共有するようになっている．これに伴い，システムの機能追加や改良等の派生開発時にシステムを跨って影響範囲を特定することが難しくなり，関連システム間の不具合が発生するようになった．DB を共有したシステムの例を図 1 に示す．関連システムとはシステム A を自システムとした場合，DB を共有して使用しているシステム B・C を指す．関連システムの不具合は，組織・部門を跨がって発生するため，不具合発生時の調査・対応工数が大きくなる．このような状態が続くと，本来実施すべき戦略的なシステム開発の工数を圧迫するなど，ビジネスに大きな支障を与

える可能性がある.そこで,DB を共有するシステム間において発生した過去の不具合事例を分析した.

当初,我々は DB に関わる不具合は，プライマリキーのような DB 構成情報を変更した場合に発生するものであると予想していた．しかし，分析の結果，DB 構成情報には変更がなく，DB に登録されるデータ項目間の関係性の変化や，データバランスの変化のようなデータ傾向が変化した場合の方が不具合件数は多いことが分かった．そして，さらに分析したところ，不具合事例に共通する発生要因と，不具合を引き起こす可能性を見過ごしやすい設計変更パターンが存在することが分かった．このパターンを利用して，DB 共有システム間の不具合発生を抑止することを考えた．不具合を引き起こす可能性を見過ごしやすい設計変更パターンを「データベース変更チェックポイント（DB-CCP)」として定義し，これを変更要求仕様と関連システム間のトレーサビリティマトリクスに組み込んだ,「変更要求仕様／関連システムトレーサビリティマトリクス（C/R-TM with DB-CCP)」を考案した.

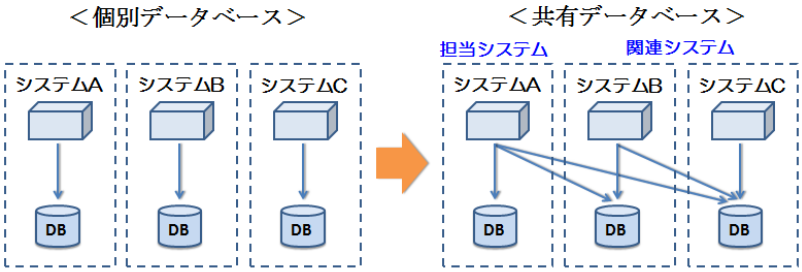

図１．データベースを共有するシステムの概念図

これにより，自システムの設計担当者は関連システムに影響を与える可能性のある変更を設計段階で検知できるようになる．また，関連システムの担当者は想定される影響を特定しやすい変更連絡を受領し，自システムの不具合発生を抑止できるようになる.

2.　データベース共有システムの現状分析

2．1.　データベースに関連する不具合事例の分析

まず，過去に発生した DB に関連する 17 件の不具合事例を収集し，分析した．不具合を「データベース構成情報の変更有無」(第 1 分類),「不具合発生箇所（担当システム／関連システム)」(第 2 分類）で分類した結果を図 2 に示す．第 1 分類の DB 構成情報の変更ありとは，具体的にはプライマリキーやデータサイズの変更等を指す.DB 構成情報の変更なし（データ傾向の変化あり）は,DB に登録されるデータ項目間の関係性の変化や，データバランスの変化を指す．変更内容の詳細を表 1 に示す.第 2 分類の不具合発生箇所（担当システム／関連システム）は，DB を共有する関連システム間において，変更を実施した自システム内で発生した不具合か，関連システムに影響を与えた不具合かを表している.

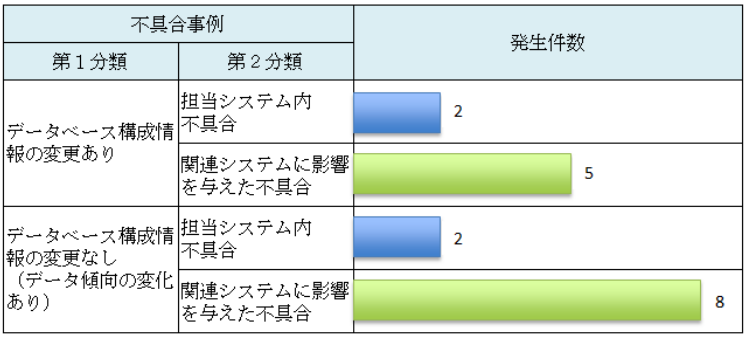

図2.データベースに関連する不具合事例の分類

表 1.データベース変更種類（具体的な変更内容）

データベース変更種類	具体的な変更内容
データベース構成情報の変更	プライマリキーの変更，データサイズ変更，カラム追加，インデックス追加等の DB 構造情報の変更
データ傾向の変化（データベース構成情報の変更なし）	DB 構成情報の変更を伴わないデータ変化,データ項目間の関係性の変化，データ量の増減，既存カラム値内でのデータバランス変化，データ登録順序・更新タイミングの変化等

当初，我々は DB に関連する不具合は DB 構成情報の変更によって発生するものであると予想していた．これは DB 構成情報を変更する場合は関連システムが使用する連携テーブル（I/F テーブル）への影響があり，その構成情報の変更が起因して不具合を発生させていると考えていたためである．しかし，今回収集した不具合を分析した結果では，DB 構成情報の変更ありの 7 件に比べ，DB 構成情報の変更を伴わないデータ傾向の変化の方が 10 件と多く，関連システムで不具合が発生する割合も 1.6 倍(5 対 8)と高く、不具合発生後の調査に時間がかかることが分かった．(付録 A として「データ傾向の変化あり・関連システムに影響を与えた不具合」事例の一部を掲載する）何故，データ傾向の変化で関連システム不具合が多く発生しているのか，さらに分析した結果，いくつかの共通する要因が存在することが分かった．不具合要因の全体像を特性要因図として図 3 に示す．

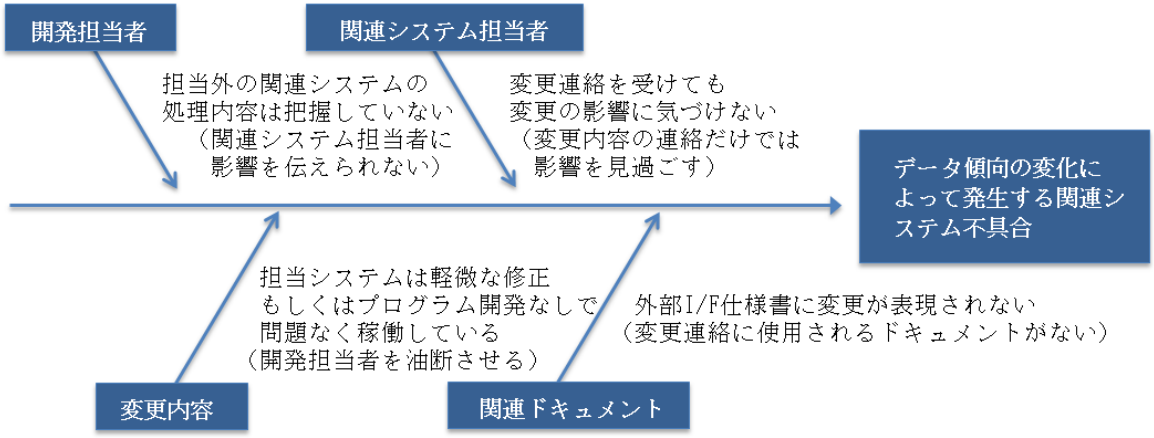

図 3. データ傾向の変更に伴う関連システム不具合の特性要因図

次に，データ傾向の変化によって発生した実際の不具合事例を表 2 に示す．

この事例では，全国に店舗展開する小売店の基幹システムにおいて，毎営業日作成される「売れ筋 TOP10 集計データ」が所定時間内に作成完了しない事態が発生した．原因は集計処理に使用している売上実績 DB（共有 DB）のデータ登録内容に変更が入ったことで，処理するデータ量が変化してしまったことにある．まず，店舗情報を管理するシステム担当者は，新潟・長野・山梨の管轄を中部エリアから関東エリアに変更するよう，システム変更要求を受領する．担当者は担当システム（店舗情報管理システム）の変更箇所を特定する作業の中で，プログラムを変更する必要はなく，店舗マスタの登録内容を変更することで対応可能と判断した．この時，担当者は関連システム担当者に変更内容を連絡したが，関連システム担当者は変更によって処理するデータ量が増加する可能性を見過ごしてしまった．

	before	after
変更要求	新潟・長野・山梨の10店舗を中部エリアから関東エリアへ変更したい	
関東エリア店舗数	20	30
中部エリア店舗数	20	10
関連システム（集計処理）	所定時間内に集計処理完了	対象店舗数増加により集計処理が完了しない

表2． 小売店の集計システムの不具合事例

これによって担当システム（店舗情報管理システム）は問題なく稼働したが，店舗マスタの管轄エリア情報が変更になったことで，関連システム（データ集計処理）で使用している売上実績テーブルの関東エリア売上データ量が大幅に増加し，集計システムが所定時間内に処理完了しない事態となった．

このように，データ傾向の変化は「開発担当者が関連システムの影響を想定できない」「関連システム担当者は変更連絡を受けても影響を見過ごす」事態を引き起こす．これは不具合の可能性を含んだ危険な状態(未病)であり，突然に重大不具合が発生(発症)する．

2. 2. 先行研究

DB 共有システム間の不具合抑止に関する問題解決として,派生開発プロセスである XDDP（eXtreme Derivative Development Process）や先行研究を適用できないか調査した．XDDP の変更３点セット(変更要求仕様書,トレーサビリティマトリクス(以降,TM と記載する),変更設計書）は担当者の思い込みによる変更漏れを抑止する効果がある[1][2]．また，システムプロファイルは予め周辺システム名称とその概要を準備することで，変更要求の作成段階での他システム影響検知を行っている[3]．しかし，DB 共有システム間の不具合は開発担当者が担当外の関連システムの処理内容を把握していないため，変更の影響を想定できないという要因を含んでいる．このため，XDDP 及びシステムプロファイルからでは不具合の影響に気づくことは困難である．また，間接リソースの変化点から影響箇所に気付く研究では，変更設計書から間接リソース変化点を抽出し，リソースに関する不具合抑止を行っている[4]．しかし，検知対象の不具合をリソース関連に限定しており，また関連システムに対する影響検知には触れられていない．このように，先行研究では DB を共有し合うシステム間で発生する不具合要因に対して効果的な対策はできていない．

3. データベース共有システム間不具合の解決策

我々は，開発担当者が設計段階で関連システムの不具合の可能性を検知できるようにすること，関連システム担当者が変更連絡を受け取った際に自システムの影響に気付けることを目指し，解決策を提案する．

3. 1. 不具合検知の解決策

何故，開発担当者は不具合の発生するデータ傾向の変化を検知し，関連システム担当者に変更連絡ができないのか，その要因を把握するために過去の不具合から変更連絡をするべきであった事例を洗い出した．そして，データ傾向の変化によって不具合が発生する設計変更パターンがあることを発見した．この変更パターンをうまく活用することで，設計時に不具合が発生する可能性のある変更かどうかを判断できると考えた．この不具合の発生する可能性のあるデータ傾向の変更パターンを「データベース変更チェックポイント(以降，DB-CCP：Database Change Check Point)」と定義した．定義内容を表 3 に示す．

表 3．DB-CCP の具体例

	DB-CCP	想定される影響
P1	他テーブルでプライマリキーとして設定されている項目の変更	・他テーブルでプライマリキーとして使用されているカラム値を変更する場合，連携システム内でのデータ集計などに影響を与える可能性がある
P2	システム利用方法（運用）の変更による値の変化	・他の項目との整合性・関係性が崩れる可能性がある 例：定義上は NULL が許容されたテーブルで，今までは NULL 値が存在しなかったが，今回運用として未使用時の NULL が送信されることになる（0:OK,1:NG→0:OK,1:NG,NULL）
P3	データ更新の処理順序の変更	・他の項目との相関関係が変更になり，エラー等が発生する可能性がある ・データ連携順の変更を考慮する必要がある
P4	データ処理件数/データバランスの変更	・リソース（データ容量，処理時間）が増加する可能性がある ・テーブル内のデータバランスの変化がおこる可能性がある

3. 2. DB-CCP での連絡，確認方法の解決策

XDDP の変更要求仕様書には変更要求の対象となる全機能が記載される．変更機能の抽出時に併せて DB テーブルの変更箇所についても抽出できるため，変更要求仕様書に DB-CPP を追加することが有効であると考えた．また，変更対象の機能と DB-CCP を同時に調査することにより作業を統合でき，煩雑さを軽減できる．次に，変更対象となる DB テーブルを管理する必要がある．１つの仕様変更につき，複数の DB テーブルが変更される可能性があることから，変更要求仕様書+TM での表現が有効であると考えた．しかしながら，XDDP の変

更要求仕様書+TM は一般にはモジュールの変更を記載するので，DB の変更を扱えるように既存のフォームに DB 変更時の必須チェック項目（DB-CCP）と，関連システムの TM を組み込んだ「変更要求仕様／関連システムトレーサビリティマトリクス（以降，C/R-TM with DB-CCP)」(図 4）を作成した．DB-CCP と TM を結合することにより，変更要求とデータ傾向の変化のあるテーブルが一覧として記載されるため，データベース構成情報に変更がない場合も，「要求」「理由（背景)」「想定される影響」の 3 点を関連システム担当者へ連絡可能となる．また，「想定される影響」を作成するための情報源となる過去の不具合事例を記録するために，C/R-TM with DB-CCP の付随シートとして「モニターシート」を作成する．

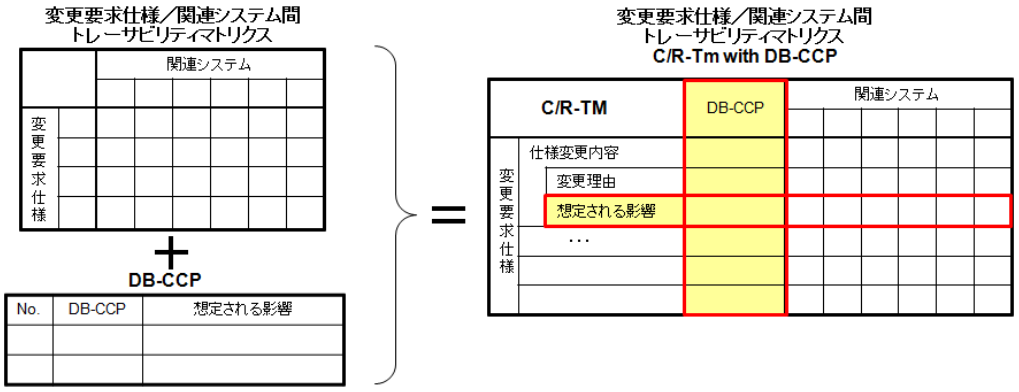

図 4.　「C/R-TM with DB-CCP」

3．3.　C/R-TM with DB-CCP の作成方法と利用方法

（１）　C/R-TM with DB-CCP の作成方法

自システムの開発担当者が設計段階で作成する C/R-TM with DB-CCP の作成手順を図 5 にて説明する．また，このテンプレートを付録 B に掲載する．

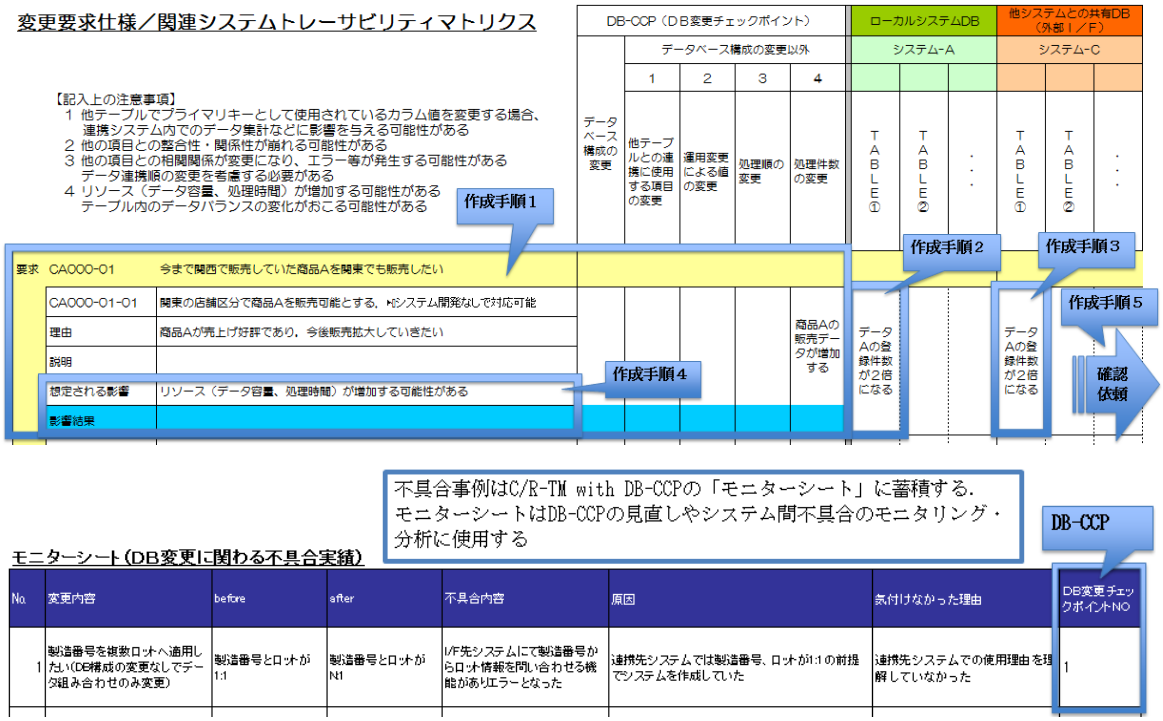

図 5.「C/R-TM with DB-CCP」「モニターシート」

【作成手順 1】
変更要求仕様書の仕様を記述する際，C/R-TM with DB-CCP の「要求」「理由（背景)」「変更内容」を記載する．記載した変更内容から，変更対象となる DB テーブルを特定してローカルシステム（担当システム）DB のテーブル列にテーブル名を記載する．
【作成手順 2】
変更対象となる DB テーブルの変更内容が，DB-CCP に該当する変更か判断し，該当する場合にはマトリクス表の該当セルに簡潔に変更内容を記載する．
【作成手順 3】
作成手順 2 で変更内容が記載されたテーブルから，ER 図や I/F 仕様書を元に，関連システムが使用する共有 DB テーブルを抽出し，他システムとの共有 DB 側のマトリクス表の該当セルに簡潔に変更内容を記載する．
【作成手順 4】
モニターシート（過去の不具合事例情報）から，DB 変更チェックポイント№に該当する不具合事例の内容を抽出し，「想定される影響」として記載する．
【作成手順 5】
変更が記載された関連システムの担当者へ,「変更内容」だけでなく「要求」「理由（背景)」「想定される影響」の 3 点の情報を含んだ影響確認依頼を行う．

（２） C/R-TM with DB-CCP の利用方法
関連システム担当者が影響確認依頼を受け取った際の C/R-TM with DB-CCP の利用手順を説明する．
【利用手順 1】
受け取った C/R-TM with DB-CCP の「変更内容」「要求」「理由（背景)」「想定される影響」から，自システムの変更の影響有無を調査する．
【利用手順 2】
調査結果を C/R-TM with DB-CCP の影響結果欄へ記載し,確認依頼者へ結果を通知する．

利用手順 1 に示すように,「変更内容」だけでなく「要求」「理由（背景)」「想定される影響」が通知されることによって，関連システム担当者は，今回の変更に関する調査が必要かどうかを推測できるようになる．仮に記載された「想定される影響」が的外れな内容であったとしても，例えば「地域によってデータ件数が増加する可能性がある」といった情報から自システムの影響を想像させることによって，過去の不具合の記憶を想起させ，調査を動機づけることができる．

3．4． C/R-TM with DB-CCP のブラッシュアップ方法
C/R-TM with DB-CCP は，各システムの特性や関連システム間のデータ使用方法が様々であることから，運用する組織の特徴が現れやすいと考えられる．いくつかの DB-CCP を表 3 に示したが，あくまでも固有システムでの一例であるため，検知能力を高めるためにはブラッシュアップが必要となる．プロジェクト管理者が公式ドキュメントとして管理担当者を決め，ブラッシュアップを行うことが重要である．1 名体制で難しい場合は 2 名で管理するなど，負荷にならないような体制で臨むことが継続的な維持，管理に有効と考える．
（１） モニターシートのブラッシュアップ
各組織で発生した DB 共有システムの不具合事例の収集・蓄積によって，各システムの特性を反映した DB-CCP の作成・維持（定期的な見直し）を行う．不具合事例は図 5 下段に示したモニターシートへ定期的に登録する．
（２） DB-CCP のブラッシュアップ
各システムの不具合特徴を反映し検知能力を向上させるために，モニターシートの

情報から DB-CCP の項目をメンテナンスすることが有効となる．メンテナンスは変更要求仕様書を記載したプロジェクトの管理者がモニターシートの記載内容から DB-CCP の項目の追加や削除の判断を行う．ただし，DB-CCP の設定上限数をあらかじめ設定することを推奨する．DB-CCP を過剰に設定した場合，チェック工数の増加に伴い，開発設計工数，関連システムへの影響確認依頼が増加する．工数の増加によって重要な影響確認依頼の見落としが発生する可能性があるため，DB-CCP の優先度を判断し上限 5 項目程度での運用開始を推奨する．

（３） 関連システム TM のブラッシュアップ

関連システム TM は外部 I/F 設計仕様書と同等の組織間の公式なドキュメントとして管理する．関連システム TM への DB 追加・更新は，他の組織（部門）が DB の使用を宣言した場合にのみ許可する．宣言がなく DB を使用した場合，開発担当者は関連システムとして認識することは困難であり，影響確認依頼を行えない．このような事態を防止するために，使用する DB を宣言したシステムのみ影響確認依頼を受け取る権利を得ることとする．

4. C/R-TM with DB-CCP の有用性の検証

C/R-TM with DB-CCP によって DB 共有システム間の不具合を抑止する効果が得られるか，過去に発生した不具合事例に適用してシミュレーションを実施した．

4. 1. C/R-TM with DB-CCP の検証方法

C/R-TM with DB-CCP の有用性を検証するために，図 2 で示した「データ傾向の変化あり・関連システムに影響を与えた不具合」（8 事例）について，C/R-TM with DB-CCP を適用して設計シミュレーションを実施する．検証に際して，下記 2 点の評価指標を設ける．

- 評価指標 A（開発担当者視点での評価）：開発担当者が DB-CCP によって関連システムの不具合可能性を検知できるか検証する．また，C/R-TM with DB-CCP の作成工数を計測する．
- 評価指標 B（関連システム担当者視点での評価）：関連システム担当者の立場で影響確認依頼を受け取った場合，不具合発生を抑止することができるか．抑止できた場合，その効果時間を合わせて評価する．

4. 2. C/R-TM with DB-CCP の検証結果

過去に発生した不具合事例に対して検証を実施した結果，C/R-TM with DB-CCP の使用によって，変更の内容とその対象 DB（テーブル）が記載され，変更の確認先も明確になった．（付録 C として検証結果の一例を示す）検証指標による評価結果は以下の通りである．

（１） 評価指標 A（開発担当者視点での評価）の評価結果

- 検知率：87%（8 事例中，7 事例）
- C/R-TM with DB-CCP の作成工数：15 分／事例

検証の結果，検証対象 8 事例のうち，7 事例について C/R-TM with DB-CCP で，関連システムでの不具合の可能性を検知できた．なお，検知不能と判断した 1 事例は，製造工程でのプログラミングミスによる単純不具合であった．

（２） 評価指標 B（関連システム担当者視点での評価）の評価結果

- 不具合発生の抑止能力：86%（7 事例中，6 事例）
- 不具合抑止によって低減できた効果時間：74 時間

検証の結果を表 4 に示す．C/R-TM with DB-CCP にて検知が可能となった 7 事例について，変更内容だけでなく「要求」「理由（背景）」「想定される影響」の 3 点の情報を受け取ったことで，6 事例については変更の影響を検知できたものと評価した．また，不具合が抑止されたことで 74 時間の工数削減効果を確認できた．

表 4.検証結果・不具合抑止件数と削減効果

単位時間：H

事例No.	不具合事例実績				C/R-TM with DB-CCP適用				
	不具合影響度	設計	開発	不具合対応	評価指標A検知結果	評価指標B抑止結果	設計	開発	不具合対応
1	重大	4	5	20	可	可	5	6	0
2	重大	7	4	20	可	可	8	6	0
3	中	6	7	15	可	可	6	8	0
4	中	4	5	15	可	可	4	6	0
5	小	2	3	7	可	可	2	4	0
6	小	2	4	7	可	可	3	5	0
7	小	2	3	7	可	不可	2	3	7
8	小	3	3	7	不可	-	3	3	7
合計	-	30	34	98	-	-	33	41	14
			総時間（H）	162				総時間（H）	88

4. 3. C/R-TM with DB-CCP の検証結果に関する考察

検証の結果，C/R-TM with DB-CCP の有用性を確認することができた．また，C/R-TM with DB-CCP の副次的な効果についても発見することができた．DB-CCP の作成・維持には不具合事例の収集・蓄積が重要である．こうしたシステム間の不具合情報の蓄積は，開発担当者の担当外システムへの理解を深める効果(オーバーラップ領域の拡大効果)が期待できる．関連システム間の不具合抑止や生産性の高いシステム開発には，こうしたオーバーラップ領域の拡大によって，相互に補完・協力する関係性が重要である．また，課題として，不具合事例の収集や C/R-TM with DB-CCP の維持に必要となる工数の負担感から，情報の最新化が行われなくなることが懸念される．この場合，DB-CCP の設定上限数を重要度によって絞り込む等の工夫で担当者の負担を低減させ，継続活用を推進していく必要がある．

5. 本研究のまとめ

5. 1. 研究成果

エンタープライズ系システムでは,複数のシステムが DB を共有することで関連システム間の不具合が発生し，突然のシステム停止によってビジネスに重大な損失を与えかねない状態となっている．我々は過去の不具合を分析し，開発担当者が設計段階で関連システムの影響を検知できるようにすること，関連システム担当者が変更連絡を受け取った際に自システムの影響に気付けることを目指し，C/R-TM with DB-CCP を作成した．今回はシミュレーションではあるが,この解決策によって DB 共有システム間の不具合を抑止する効果が期待できることが分かった．

5. 2. 今後の進め方

今後は実際の派生開発の現場にて C/R-TM with DB-CCP の効果を検証し，さらに改良していく．特に，C/R-TM with DB-CCP の組織知蓄積の効果を実証し，ビジネスに重大な影響を与えかねない関連システム間の不具合抑止に貢献していく．

6. 参考文献・参考情報

[1]清水吉男,「派生開発」を成功させるプロセス改善の技術と極意，技術評論社，2007

[2]清水吉男，[入門＋実践]要求を仕様化する技術表現する技術，技術評論社，2010

[3]木下良介，中澤康郎，大杉仁司，変更依頼の対応箇所を検討する前に他システムへの影響を検知する方法，日本科学技術連盟　ソフトウェア品質研究会分科会報告書，2011

[4]小瀬聡幸，衣斐省伍，井貝智行，杉山幸雄，XDDP の変更設計書から間接リソース変化点を抽出する手法，日本科学技術連盟　ソフトウェア品質研究会分科会報告書，2013

[5]データベースに関するシステム障害事例：楽天証券公式 HP，楽天証券・2008 年 11 月 11 日発生不具合，https://www.rakuten-sec.co.jp/web/info/info20090116-04.html

付録 A： データベース共有システム間における不具合事例（データ傾向の変化に起因する関連システム不具合）

No.	変更内容	before	after	不具合内容	原因	気付けなかった理由	データベース構成情報
1	製造指示を複数ロットへ適用したい（データ組み合わせのみ変更）	製造指示とロットが 1:1	製造指示とロットが 1:N	I/F 先システムにて製造指示からロット情報を問い合わせる機能で想定より少ない表記となった	連携先システムでは製造指示，ロットが 1:1 前提の機能が存在した	連携先システム（他社）での使用理由を理解していなかった	変更なし
2	同一ロット番号を複数設備で製造するように変更	ロットは設備間をまたがない	ロットが設備間をまたぐ	連携先システムでは設備毎実績の値が合わなくなった	連携先システムでは設備の情報をロット番号内の桁数で判定していた	連携先システムとの設備判定のルール違いを認識していなかった	変更なし
3	判定要素追加に伴い，既存判定要素の使用区分で NULL が設定される	使用区分は数値のみの登録	区分は数値と空白で登録される	I/F 先システムにてデータ数が合わなくなった	連携先システムにて PK ではない使用区分を利用してデータ集計を行っていた	連携先システムでの使用理由を理解していなかった	変更なし
4	既存工程を通らずに新工程のみを行うように運用を変更	A→B	A→B，Bのみ	連携先システムの画面起動が異常に遅くなった	Bのみデータの取得を行う仕組みではあったが，Aの結果が NULL のデータが大量に増えた時に INDEX が効かずデータ集計処理で約 50 倍の時間がかかるようになった	大量データ内容変更時のテスト不足	変更なし
5	ユーザ部門の要望を実現するために，ある画面で稼働しているDCの処理判定順序を変更した.	項目 A の更新タイミングは登録画面で行われる	項目 A の更新タイミングは登録画面後の後続画面で行われる	後続のバッチシステムでエラーが発生した	当該DCは想定通り稼働しているが，処理順を変更したことにより，別処理で使用した項目の更新前データが誤って使用されてしまった	処理順変更による，データ更新状態の取り違え.	変更なし
6	オンライン画面側で入力項目の桁数チェックを実装した	項目 A は桁数チェックなし	項目 A は桁数チェックあり	連携システム側で桁落ちが発生した	後続の連携システムでの有効桁数を考慮できていなかった	連携先システムでの使用理由を理解していなかった	変更なし

7	照会受付システムの中である処理を，商品 A に向けに実装した．これを商品 B にも使用したい	商品 A のみでシステム使用	商品 A･B でシステム使用	ロジックは正常に稼働したもののレスポンス悪化の原因となってしまった．	処理件数の増加によるリソース負荷を考慮できていなかった	処理件数増加による後続システムの影響を考慮していなかった	変更なし

付録 B： C/R-TM with DB-CCP

変更要求仕様／関連システムトレーサビリティマトリクス

【記入上の注意事項】
1 他テーブルでプライマリキーとして使用されているカラム値を変更する場合、
連携システム内でのデータ集計などに影響を与える可能性がある
2 他の項目との整合性・関係性が崩れる可能性がある
3 他の項目との相関関係が変更になり、エラー等が発生する可能性がある
データ連携順の変更を考慮する必要がある
4 リソース（データ容量、処理時間）が増加する可能性がある
テーブル内のデータバランスの変化がおこる可能性がある

		DB-CCP（ＤＢ変更チェックポイント）					ローカルシステムDB			他システムとの共有DB（外部Ｉ／Ｆ）		
		データベース構成の変更	データベース構成の変更以外				システム-A			システム-C		
			1	2	3	4						
			他テーブルとの連携に使用する項目の変更	運用変更による値の変更	処理順の変更	処理件数の変更	TABLE①	TABLE②	・・・	TABLE①	TABLE②	・・・
要求	A000-01											
	A000-01-01											
	理由											
	説明											
	想定される影響											
	影響結果											
	A000-01-02											
	理由											
	説明											
	想定される影響											
	影響結果											

モニターシート（DB変更に関わる不具合実績）

No.	変更内容	before	after	不具合内容	原因	気付けなかった理由	DB変更チェックポイントNO

付録Ｃ：　C/R-TM with DB-CCP 検証結果例

変更要求仕様／関連システムトレーサビリティマトリクス

【記入上の注意事項】
1 他テーブルでプライマリキーとして使用されているカラム値を変更する場合、
連携システム内でのデータ集計などに影響を与える可能性がある
2 他の項目との整合性・関係性が崩れる可能性がある
3 他の項目との相関関係が変更になり、エラー等が発生する可能性がある
データ連携順の変更を考慮する必要がある
4 リソース（データ容量、処理時間）が増加する可能性がある
テーブル内のデータバランスの変化がおこる可能性がある

			DB-CCP（ＤＢ変更チェックポイント）					ローカルシステムDB			他システムとの共有DB（外部Ｉ／Ｆ）		
				データベース構成の変更以外				システム-A			システム-C		
				1	2	3	4						
			データベース構成の変更	他テーブルとの連携に使用する項目の変更	運用変更による値の変更	処理順の変更	処理件数の変更	エリアマスタ	店舗マスタ	売上実績	店舗毎売上実績	エリア別売上実績	・・・
要求	CA000-01	長野、山梨、新潟を関東エリアへ変更する											
	CA000-01-01	中部エリアの県コード長野、山梨に所属する店舗のエリアを関東へ変更する					関東エリアの処理件数増加	エリア名変更	長野、山梨の事業所営業区分を5→3へ変更			関東エリアの処理件数増加	
	理由	関東エリアを関東甲信越エリアとし、長野、山梨、新潟を組み込んで管理を行うため											
	説明	事業所所在値コード23：新潟、24：長野、25：山梨の事業所営業区分を5：中部から3：関東へ変更する											
	想定される影響	ローカルシステムでの影響なし I/F先で店舗マスタのエリアで集計を行っている場合は処理時間に変化がある可能性あり											
	影響結果	関東のエリア内店舗が30店舗増加。エリア別集計時間が1.3倍に増加する可能性があるため、処理開始時間をＡ処理と同タイミングに変更したい。											