モデリングによる暗黙知分解とスキル補完への取り組み
～共感と共創をつくり，人材不足解消と多能工を促進～

三輪　東　　清田　和美
SCSK 株式会社
azuma.miwa@scsk.jp
kazumi.kiyota@scsk.jp

與儀　兼吾
SCSK ニアショアシステムズ株式会社
kengo_yogi@scsk-nearshore.co.jp

日下部　茂
長崎県立大学
kusakabe@sun.ac.jp

要旨

大規模システムを広範囲に担当している現場では，複数の異なる作業や工程に対応できる多能工人材の存在は貴重であり，我々は積極的にその育成を行っている．そのような多能工人材の育成には，複数の異なる作業や工程の経験が必要であり，次のようなことが重要と考えている．長年の経験などで積み上げられたルールや作法が暗黙知となり第三者から分かりにくい点を分解・可視化し新規参入者への理解を容易にする，関係者のフォローや協力によってスキル不足を補完する合意形成をする，などである．

本経験論文では，システム理論に基づく事故モデル STAMP とその分析手法である STPA や CAST が，暗黙知の理解を助けスキル補完の合意形成を促進し，多能工を促進する組織のツールとして役立った一例を報告する．

1. はじめに

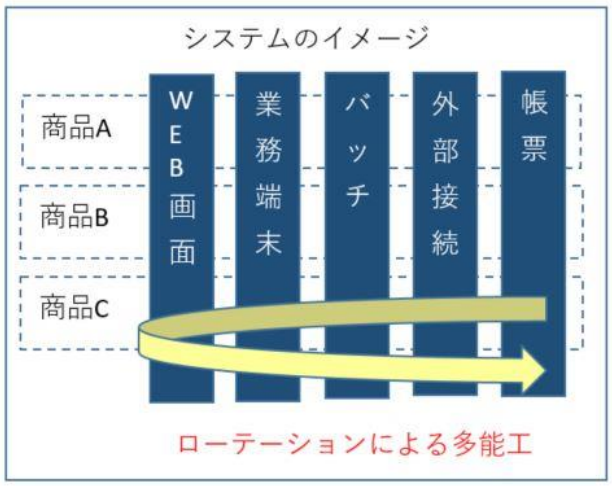
1.1. 継続保守・開発，大規模ゆえの中期人材育成

我々は基幹系システム全体，アプリケーション・インフラ・運用の開発・保守を担当している．開発案件の担当工程は立ち上げから導入まで全て，保守は 24 時間 365 日である．担当領域が非常に広く，様々な知識と技術を必要とする．毎月なんらかの改修を行っている機能もあれば，1年に1回改修が入るか入らないかの機能があるなど，頻度や仕事量は一定ではない．法令改正や新サービス追加で，ある時期だけ仕事量が急激に増えることもあり，要員配置の苦労は尽きない．そのような中で，様々な仕事に適応できる多能工人材の存在は，人材不足解消・ノウハウ維持・生産性と品質維持など多数の有益な価値を生むため，積極的な育成を委託先とも協力しながら行っている．

1.2. 本経験論文の多能工

多能工人材のパターンは一つではないが，本経験論文における多能工人材とは，WEB 画面・バッチ・外部接続機能など，複数サブシステムの開発工程を担当できるエンジニアのことを指す．図1．では，5つのサブシステムを持つシステムを示している．2 つ以上のサブシステムで開発を行える人材を，多能工人材とする．

図 1. 本経験論文における多能工前提



1.3. 本経験論文のねらい

本経験論文では，そのような多能工推進の中で発生した，暗黙知とスキル不足を要因とする事故の分析に STAMP 手法を活用してモデリングした事例を通じて，以下を考察する．

STAMP モデリングが，
Q1. 暗黙知の分解にどのように役立ったか

Q2. スキル不足の補完にどのように役立ったか
Q3. 共感と共創に役立つものか
Q4. 多能工に好影響を与えるのか

あわせて本事例の STAMP 手法を用いたモデリングの経緯を報告することで，同手法やモデリングの導入に役立つ情報提供を目指す．

2. 多能工促進方法と課題

2.1. 多能工促進の方法

多能工人材を育てる一般的なアプローチは，あるサブシステムで経験と実績を積んだメンバーを他チームへローテーションさせる方法である．ローテーション前後の開発言語は同一とは限らない．サーバアプリとクライアントアプリでは開発環境も異なる．オンラインとバッチでは設計に対する基本的な考え方も同じではないなど，様々な違いがある．別のサブシステムで一定の成果を出してきたメンバーやリーダークラスであった場合でも，ほぼゼロからのスタートというケースも少なくない．

なお，プロジェクト管理方法・課題管理方法・レビュー実施における必須成果物群・他サブシステムと協業で作成する成果物群などは全プロジェクト共通だが，プロジェクトで管理するタスクの詳細やタイミングなどはサブシステムの裁量に任せられており必ずしも共通ではない．

2.2. 多能工促進の課題

多能工推進を組織として明示的に掲げてから 5 年以上になるが，経験則から問題となりやすい点が二つあると感じている．一つは暗黙知の問題．もう一つが暗黙知と合わさった複雑なスキル不足の問題である．今回は，この二つの課題に焦点をあてる．

2.2.1. 暗黙知が新規参入を難しくする

本経験論文では暗黙知を次のように定義する．「新規参入者には理解しにくい，新規参入者を受け入れたチームではあたりまえの知の集合体と現状．ルール・規律，言語パッケージやフレームワーク構造，作法・判断基準など，普段は言及されないもの全般を含む．」 例えば，外部接続先とのインターフェース仕様書に記載されている情報は暗黙知には含まないが，その設計思想や，様々な実装者が修正していったソースコードの経緯など，知の集合体を背景に持つファイルそのものと，そのファイルを取り扱う前提となる様々な作法や期待される行動などの全般を指す．

暗黙知に起因する問題の一例を示す．新規参入者が新しいチームのルールや作法が分からず，過去所属していたチームのご作法で成果物作成を行った結果，そもそもの前提が誤っていたためチームの期待に応えられないケースがある．例えば，成果物の求める達成レベル（早く出してチェックを繰り返す，じっくり最後でチェックする違いなど）の考え方が異なり，双方が期待したタイミングで成果物が作成されない状況などである．受け入れたチームが早く出してチェックを繰り返すスタイルにもかかわらず，参入者がじっくりと時間をかけてしまう場合，新規参入者が不慣れであることを考慮すれば，初回のチェックで十分な結果が得られることは予想しにくい．結果，余計な時間をかけたことによるコストや納期への影響，納期ひっ迫原因による品質への影響などの悪い要因を生むことは容易に想像できる．逆にチームがじっくり最後でチェックするスタイルで参入者は早く繰り返しスタイルの場合，チームメンバーが「新規参入者にはこの程度のスキルしかないのか」と誤解し，新規参入者の今後の協業に様々な支障が出る可能性もある．いずれにせよ，プロジェクト QCD に何らかのかたちで影響し，受け入れチームと新規参入者の双方にストレスになることは間違いない．

問題を複雑にするのは，こういった暗黙知は長年チームで蓄積されたノウハウや認知的側面[2] が多いことによる．SECI モデル[3] のように歴史と共に変化・進化し，プロセスや成果物フォーマットなどにちりばめられつつ，その現場の慣習や文化になったものは，可視化が難しい．7S の「ソフトの 4S（Shared value , Style , Staff , Skill）」[4] に染み渡った状況であり，プロセス資産，個人のスキルや意志・行動，文化といった組織の環境要因に相互依存しながら混じり合い，境界線を引くことも難しい．そこにいる当事者であっても，その構造を端的に説明することは難しい場合も多い．受入チームメンバーが言語化出来ないものを，新規参入者が簡単に理解できるはずもなく，問題をより複雑化させる．

文献[2] (野中郁次郎と竹内弘高の著書『知識創造企業』，梅本勝博訳，東洋経済新報社 P9）では，暗黙知の認知的側面を次のように述べている．

> 暗黙知には重要な認知的側面がある．これに含まれるのが，スキマータ，メンタル・モデル，思い，知覚などと呼ばれるもので，無意識に属し，表面に出るこ

とはほとんどない．この認知的側面は，我々が持っている「こうである」という現実のイメージと「こうあるべきだ」という未来へのビジョンを映し出す．簡単には言い表せないこれらの暗黙的モデルは，我々が周りの世界をどう感知するかに大きな影響を与える．

2.2.2. 十分条件にならないスキル要件

例えば Java での開発業務にもかかわらず Java 自体を知らないような純粋なスキル不足はここでは扱わない．現場で発生しやすいスキル不足の問題は，必要条件としては整っているものの十分条件まで満たされておらず，問題になるケースである．

例えば，SQL を使って開発しているオンラインアプリとバッチアプリがある．どれも使う SQL 構文は変わらない．しかし，オンラインアプリであれば，ロックを最小限にする工夫を求められたり，バッチであれば短時間に大量データを処理ことが必須であったりする．非機能面の要求は「SQL」という言語記号だけでは表現することが出来ない．長年継続開発・保守を担当しているチームでは，非機能要件を満たすための工夫が暗黙知になっており，全体の構造まで明示的に説明されないケースも多い．事前に「性能を意識したコーディングをして下さい．」と伝えても，具体的な How が暗黙知となっており，新規参入者に正確に伝わらず，受入チームの期待に応えられないケースもある．

Java のように，たくさんの機能拡張されている言語分野でも問題が発生しやすい．「Java を使います」では Java EE や J2EE のどの機能まで必要なスキルになるのかを十分に説明出来ていない．開発環境はどうなっているか，自動テストをどのように行っているかなどでも同様のことが起こる可能性がある．受入チームからすればあたりまえの事実になってしまっていることが事前に説明されず，開発をスムーズにスタート出来ず，問題となる場合もある．

3. モデリング対象選定の経緯

あるチームで発生した事故の原因分析と再発防止策策定を対象にモデリングを行った．この事例では関係者が納得のいく再発防止策をなかなか策定できず，困り果てていた．

事故の直接原因は「ソースコードの不適切な修正とテスト未実施による検出漏れ」であり，その報告だけを聞いた段階では，モラルハザードの発生が疑われた．しかしながら，詳細なヒアリングにより背景にはスキル不足があり，加えて，次の気になる点があることが分かった．

3.1. 指示・受入側もスキル不足は理解しており，フォローに対して積極的であったにも関わらず，事故が発生してしまった．
3.2. 作業実施側も何とか期待に応えようと努力しており，チェックリストやレビューも実施済であったが，事故になってしまった．
3.3. 双方，やるべきことを適切にやっていたと信じており「まさかこんな事故になるとは」という気持ちと共に，自信を失っていた．

ヒアリングを通じて，現場から感じられたのは真剣さと善意であり，開発現場でモラルハザードが起きていないことは，すぐに理解できた．委託先も関連しており，慎重に対処する必要がある．委託先とは長年良い協業を続けていることもあり，実態を明らかにして，今回の失敗を教訓としてより良い状況に発展させたいと考えた．

そのため，納得できる再発防止策を策定してもらおうと，当事者で会議を繰り返してもらったが，お互いが納得できる解決策まで得られず，これまでとは異なったアプローチを必要としていた．

このような背景の下，今回の事故の特徴と STAMP の相性が良いと感じ STAMP を選択した．今回の事故は関係者も「まさか」と感じる意外性を持つ事故であった．原因も一つの単純な問題から派生したものではなく，いくつかの相互に依存する問題が重なり合った結果として事故になっているように感じた．これが STAMP の事故（望んでもないし計画もしてない損失につながり得るイベント），ハザード（環境のある最悪な条件の集合と重なると事故になり得る，システムの条件もしくは条件の集合）[5] (P16) と非常に似通っていると感じ STAMP を採用した．

4. モデリング経緯と結果

4.1. 登場人物

事故が発生したプロジェクトのチームの登場人物は次の通り．

① 指示・受入管理者：全体責任者

② 指示・受入責任者：実務遂行する責任者
③ 作業実施管理者：委託先責任者
④ 作業実施責任者：作業実施責任者
⑤ レビューア：主に設計書・ソースコード・テスト結果レビューを担当
※作業実施責任者が兼務する
⑥ 作業担当者：作業実施者の指示のもとで，設計書修正，コーディング，テスト実施

表1 登場人物のスキル

	所属	スキル充足度	特記事項
①	委託元	問題なし	
②	委託元	問題なし	
③	委託先	問題なし	
④ ⑤	委託先 （兼務）	問題あり 新規参入者	技術要素が異なるローテーションで不慣れ
⑥	委託先	問題なし	スキルに問題はないがミス多い

4.2. モデリングの流れ.

以下の順序でモデリングを行った.
4.3. モデリング前の再発防止策を整理
4.4. 初回モデリング
4.5. 初回 CAST アプローチ
4.6. 経緯と歴史の紐解き
4.7. CAST アプローチ再実施
4.8. STAMP モデリング実施

4.3. モデリング前の再発防止策を整理

モデリング開始前に委託先，委託元の関係者を集めてヒアリングを行った．獲得した情報は以下の通り．「→」に続く個所は考察を含む．

a. モラルハザードはない．
b. 言いたいことは言える関係がある．
c. 委託元作業指示以外の作業は行わないルール．
※c. は事実だが一部情報が欠落，後の過程で情報が追加される．
d. ソースコードの不適切な修正は，作業担当者の思い込みによるミスで，指示・受入責任者が指示した箇所以外を不用意に修正してしまった．
→ ミスは誰でもあることなので完全には防げない前提で考え，レビューやテストの安全制約があることを確認した．
e. テストが未実施
→ 役割分担としてやるべきことが出来ていない，プロセス違反（なぜ？）
※e. この段階の見解で事実と異なる．後の過程で見解が変わる．
f. レビューア d., e. の欠陥を，レビューで取り除く役割だが，指摘出来ず．スキル不足が背景にある．
※f. この段階の見解で事実とは異なる．後の過程で見解が変わる．
g. レビューアは多能工推進で別サブシステムからローテーションされたメンバーで経験が浅い．
→ 多能工は双方合意の事項であり，e.を発見できなかった原因である f. が問題とは単純に言えない点で，双方に相当のストレスがある．
h. DIFF による対象範囲外修正の有無は委託先で行っており，実施したが「問題なし」と判断してしまった．委託元（受入側）では実施済の確認をいつも通り行っていた．
→ 長年の経緯で，数年前から委託先で行われることになっている．これまで事故はなかった．委託元でもチェックすべきではないか？
i. 再発防止策として，委託元でも委託先と同様の DIFF チェックを行うように変更．

4.3.1. モデリング開始前の関係者心境

この時点の関係者の心境を纏めておく．

委託元には委託先の不満が溜まっている．関係者がやるべきことすら普通にやれない状況と理解しており，何を改善すべきかと途方に暮れている状況だ．

委託先では，決められたルールが守れない自己嫌悪と，不慣れなメンバーがいる中での失敗をどう捉えるべきなのかの迷いがある．双方，どこかで納得できない部分があり，解決に向かうにしても，何に焦点をあてて対話していけば良いか分からない状態である．

再発防止策で，委託元で確認するプロセスを追加（i.）したことで，ひとまずの解決策までは出来上がり，顧客へ

の説明も出来たが，委託元が委託先に抱く不信感は根強いものになってしまった．委託先も自信を失っており状況は好ましくない．長年の良い協業関係を今後も続けていくためには，双方が納得できる現状分析が必要である．

4.4. 初回モデリング

ここまでの情報でモデリングしたものを，図 2．初回モデリングに示す．

この段階では，ヒアリングや当時の成果物チェックで得た情報から，安全制約は何かに着目し作成した．結果はシンプルなものとなった．しかし，STAMP/STPA 導入を決めた際に着目したハザードが捉えられていない点で，期待どおりとは言い難い．モデリングは 1 回で全て出来上がるものでは無いので，少し視点を変えて臨むことにした．

初回モデリングでは安全制約に着目したので，次は CAST アプローチを用い，ハザードと思われる状況を現場目線からのボトムアップで確認することにした．

特に以下の 3 点に着目した．

4.4.1. e. の問題がなぜ起きているのか不明
4.4.2. f. の配置は適切だったのか
4.4.3. なぜ，g. であることは理解していながらも f. の可能性を予想できなかったのか

4.5. 初回 CAST アプローチ

まず 4.4.3.「なぜ，g.であることは理解していながらも f. の可能性を予想できなかったのか」を紐解くためにヒアリングを行い，以下 c.「→」以降の追加情報を得た．

c. 委託元作業指示以外の作業は行わないルール
→ 委託先からのフィードバックや問い合わせをもとに作業指示に追加される場合がある．
委託元はフィードバックの有無で，作業指示の妥当性を確認している側面がある．

追加情報から派生して j. を獲得した．

j. 受入まで問い合わせが無かったので，適切に理解できていると思っていた．「何か分からなければ聞いて下さい」と伝えてあり，これまでそのスタンスで問題なかった．

「聞いてくれるものだ」との期待が j. にあり，これまでのベストプラクティスであることがうかがえる．なぜ，この期待に沿えないのかを安全制約の観点から考えてみる．

委託元には暗黙の安全制約（分からないことは聞いてもらえる）が存在していることになるが，委託先ではこの安全制約を適切に運用できていない状態と言える．コミュニケーション問題の有無については，b. からも双方良好なコミュニケーションは確立できていると認識しており，会議

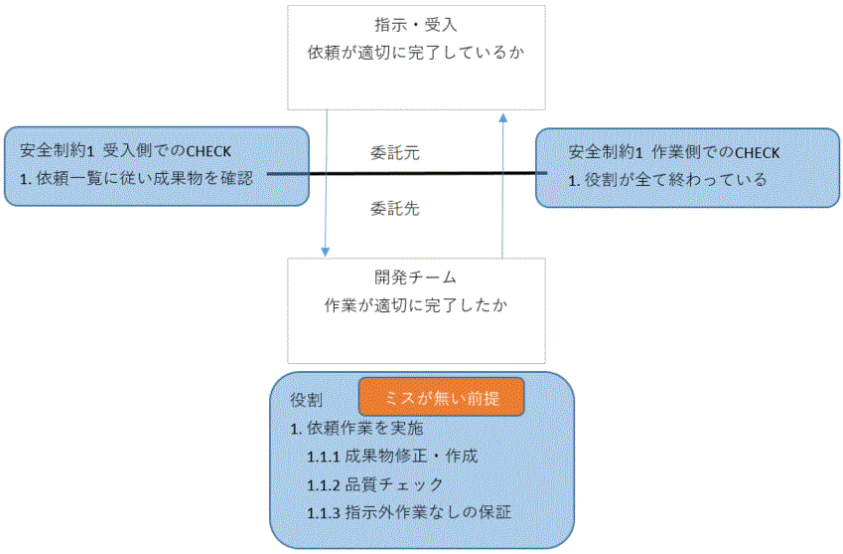


図 2．初回モデリング

の発言などを第三者がみても，心理的な障壁などで対話にならない障壁があるとは感じられなかった．

次のような仮説を立てた．「何を知っている必要があるか，何が期待されているのかといった前提の暗黙知を知らず，分かっているか否かを感知できていない，そもそも理解していない状況」と仮定した．片方は期待している一方，片方は期待を理解するレベルではなく期待の存在すら分からないとすると，双方がかみ合うことは難しい．こういった暗黙知を前提とする安全制約のずれに着目することで，真因にたどり着けると考えた．さらなる考察のために少し視点を変え，これまでの経緯をさかのぼってみることにした．

4.6. 経緯と歴史の紐解き

暗黙知を満たす期待を紐解くため，これまでの経緯，委託先と取引を開始した当初の約 10 年前までさかのぼって考えてみたのが k., l., m. である．

k. 協業開始当初，委託先リーダーは委託元の個別指導とともに仕事を一通り覚えた経緯がある．そこで，プロセス・成果物・開発環境など見えやすいものと，必要なスキル・コミュニケーション方法・判断基準などの見えにくいものを獲得し，委託先へ戻った．
l. 委託先が仕事を覚える過程で，「これで良いですか？」という細かな問いが頻繁に繰り返された．双方のフィードバックで暗黙知を築き上げていくコミュニケーションがあった．その中で，委託元は伝えられているか，委託先は理解しているかを繰り返し確認しながら，伝わりにくい部分をドキュメントに残すようにし，齟齬が発生しやすいタイミングで会議体を設定するなど，双方でより良い仕事の仕方を構築しながら，互いの暗黙知を築き上げたと言える．SECI モデル[3]の実践である．
m. 委託先メンバーが仕事を覚え，委託元と同じ基準で仕事ができる段階になり，仕事の役割分担を委託先に徐々に移譲していった．状況ごとの判断基準などの期待も含めて，委託元が理解していることが仕事の前提になっており，双方の暗黙知になっていた．

k., l., m. を SECI モデル[3] で再考する．SECI モデルは，共同化，表出化，連結化，内面化を繰り返す．そこには対話，形式知の結合，行動による学習，場作りがある．([2] の「図 3-4 知識スパイラルとその内容の変化」が分かり易い) k., l., m. を通じて共同化があり，「問い」を通じて表出化されている．その中で，どのように協業していくのがベターなのかを連結化し，成果物やプロセスという内面化を通じて個と組織にナレッジを蓄えていく．この暗黙知をベースに品質と生産性を作りこんでいる．

現状の c., j. との違いを考えてみる．c., j. は共同化と表出化が弱くなっていると言える．「知識創造企業」[2]（P360）では，トップダウンは連結化と内面化に焦点が当たり，ボトムアップは共同化と表出化に焦点が当てられていると述べられている．この考えを取り入れると，ボトムアップの取り組みが弱くなっていると言える．かつてはこの両極を統合するミドル・アップダウン・モデル[2](P360) だったと言えるだろう．暗黙知の共有が崩れたことで，そのバランスが崩れた状態と言える．

過去の過程を振り返る中で，もう一つ明らかになったことがある．h. の情報を分解した結果だ．

n. DIFF による対象範囲外修正の有無は，委託先と協業を始めたころは指示・受入担当者が行っていたが，最近の数年間は委託先に移譲しており事故は発生していない．

モデリング前に，DIFF による対象範囲外修正の有無確認を委託元が行っていないことに，やるべきことをやれていないのではとの違和感があった．実際には，m. から過去の経緯で役割を委譲したことが分かり，以後も事故は起きていないことから，しばらくは問題が無かったと言える．それが意図した運営が機能しなくなるにつれ，「より高いリスクに向かう経時的な推移 –例えばより高い効率や生産性の追求で何かを軽視」[5] (P8) に近い状況に遷移してしまったことが理解できた．当初は暗黙知とスキル充足でコントロール出来ており，生産性追求のために役割を委譲したが，経年の体制変更や仕事量の増加などで，コントロールが不十分になったと言える．

ここで難しいのは，意図的にルールを変え，高いリスクに向かう経時的な推移を推進したわけではないことだ．前提とした暗黙知が機能しなくなったことで，状態がハイリスクに遷移したとも言える．この件は課題とし，本件のモデリングからは除外することにした．

4.7. CAST アプローチ再実施

ここまでの過程で理解できたことを整理する．スキル不

足とは一般的には個人に依存するように語られるが，実際には組織構造からくる暗黙知不理解といった，安全制約をコントロールできないスキル不足も存在すると言える．であれば，知らないことを個人の問題としてとらえるのではなく，構造の問題としてとらえることが必要だ．「何が／誰が悪い？」ではなく「なぜ／どのように？」コントロール出来ていない状態なのかを追求することが必要である．この分析に強いのが STAMP の特徴 [6] (P8) だ．その視点で，再度，インタビューを行い，以下の 3 つの疑問を解決する解を得られた．

4.4.1 e. の問題がなぜ起きているのか不明
4.4.2 f. の配置は適切だったのか
4.4.3 なぜ，g. であることは理解していながらも
f. の可能性を予想できなかったのか

o. e. の原因は，テストは出来ていると作業担当者が勘違いした，が正しい．修正モジュールは画面から必ず呼び出されるものと勘違いし，画面テストは実施済なので問題なしとした．しかし，該当モジュールは画面から直接は呼び出されないコードで，別の方法でのテストが必須であった．プロセス違反ではなく，思い込み・勘違いによるミスであった．

p. e. のもう一つの原因として，レビュー時に問題を検知出来ていたが，対処が為されなかった事実が確認できた．レビューアは作業者に対し，d. 指示箇所以外の修正の妥当性確認を指摘しており，レビューアとして機能している．その指摘を作業担当者が c.「→」以降のケースもあるため，問題ないと回答してしまったことが一因である．当初の様子を，「不慣れな私でもおかしいと思ったが，私よりも経験の長い担当者が『問題ない』と言ったことで，担当者の方が正しいと判断してしまった」という背景があったことを理解できた．

q. レビューアは他サブシステムの一定実績を評価されローテーションされており，o. の実績からもレビュースキルは保持している．レビューアが多能工のローテーションから経験期間が短く，技術スキルは十分と言い切れないことを，委託元も把握済，要フォローのステータスであった．その為，本作業をお願いする前に，修正箇所を事前にチェックし，実装難易度も高くないものに限定して仕事を卸した経緯があった．その為，技術的な問題は発生しないと判断．何かあった場合は，j. のとおり問い合わせが発生する想定で可能性は予見していた．

以下にまとめを述べる．

4.4.1. o. から思い込み・勘違いによるミスと分かる．
→ ミスは誰でもあることなので，完全には防げない前提で考え，レビューやテストの安全制約があることを確認した．
4.4.2. p., q. から最適とは言えないが，最善であったと言える．
4.4.3. 委託先と委託元ともに予見していた．

4.8 STAMP モデリング実施

これまでの経緯から，再度モデリングしたものが，図 3. STAMP モデリングである．

4.8.1 モデリング実施後の関係者心境

初回モデリングから STAMP モデリングに行きつくまで，問題の抽出とそれらの相互依存関係を明らかにするため，かなりのヒアリングを要した．結果としてプロセス上省略した箇所はないことが分かり，関係者の中で暗黙知に起因したハザード（環境のある最悪な条件の集合と重なると事故になり得る，システムの条件もしくは条件の集合）[5] (P16) に起因していると合意形成できた．ミスを前提としたレビューなどの安全制約でいつもなら防げるものが，偶然にも複数の状況が重なり発生した事故であることが共有でき，事故の複雑性を関係者が理解できたことで，不信感が消えていった．長年の良い協業関係を今後も続けていくため，難しい問題に立ち向かう気概が出てきたことがとても大きい．

結果として，本件をケーススタディとしながら，ふりかえりや勉強会を実施し，暗黙知を意識しながらプロセス改善やコミュニケーション改善を行っていくという再発防止策を取り入れることが出来た．

5. 考察

「1. はじめに」で設定した，STAMP モデリングに関する Q1～Q4 の考察を述べる．最後に STAMP モデリングの感想を述べる．

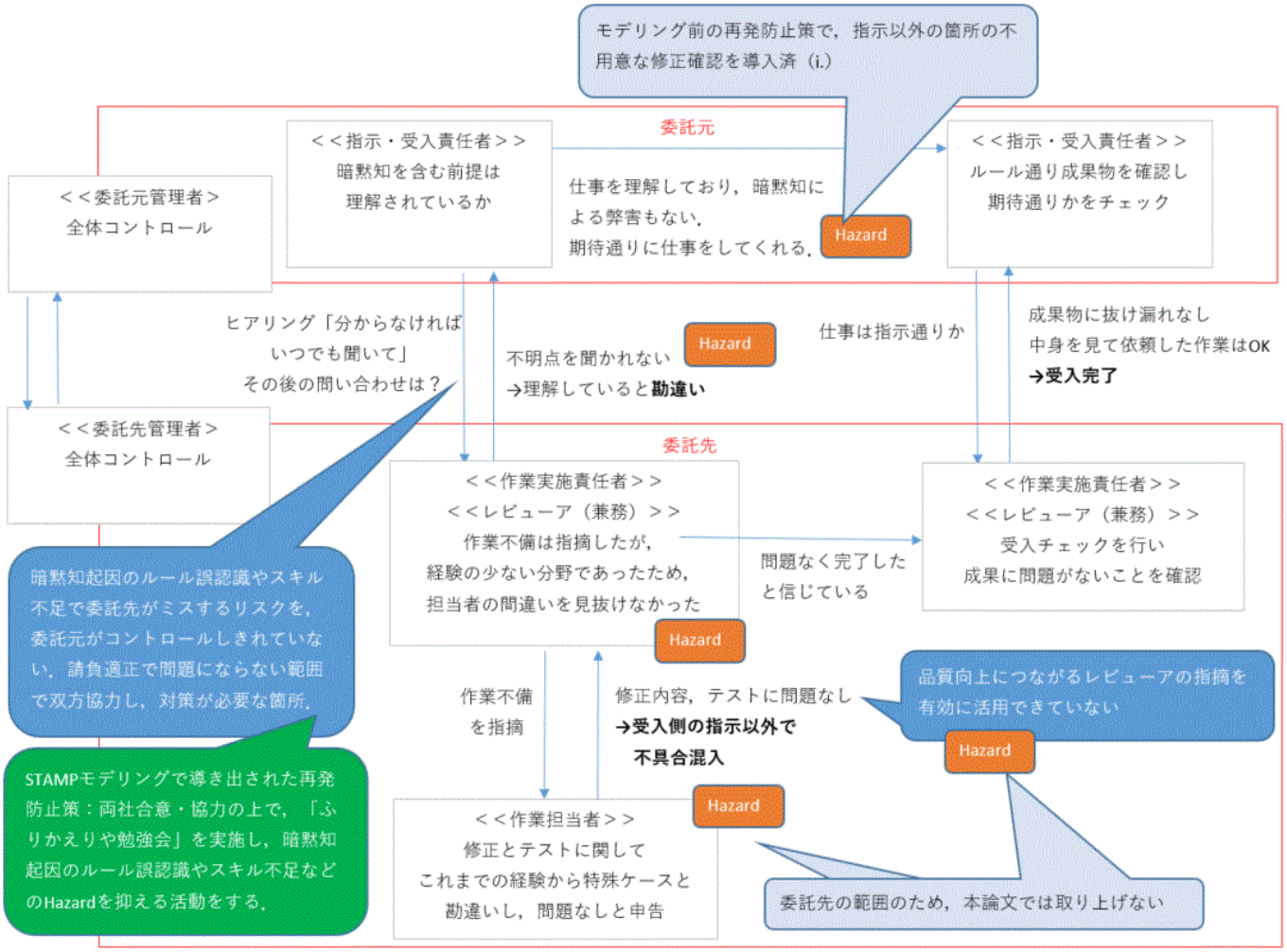


図 3. STAMP モデリング

5.1. Q1.暗黙知の分解にどのように役立ったか

暗黙知分解は時間軸をさかのぼり，かつ複数の関係者がどのように共同化，表出化，連結化，内面化を繰り返していたかを可視化する必要がある．一方向的な思考だけでは捉えることが難しい．そこで STAMP の特徴である非線形的なループ構造，コントローラーと被コントロールプロセスという視点で着目していくことが役立った．相互依存で関係者がどのように合意形成したのか，個人の成長とともに組織にどのようなドキュメントやプロセスを残しながら暗黙知として品質や生産性を上げていったのかを分解していく思考に役立った．

今回行った，情報収集→初回モデリング（ラフスケッチ）→CAST→暗黙知分解→CAST→STAMP の流れで考えていくと，暗黙知分解を効率よくできることが分かった．特に，暗黙知分解→CAST→STAMP と続くことで，相互依存関係を明記する必要があり，「なぜ／どのように？」コントロール出来ないかを書かなければモデリングが終われない．暗黙知が存在するだけで終わらせず，暗黙知がコントロールに及ぼす影響まで可視化される．この点で STAMP/STPA は強力なツールと言える．

5.2. Q2.スキル不足の補完にどのように役立ったか

組織全体にちりばめられている暗黙知を把握することで，構造からくる暗黙知不理解といったスキル不足が存在することを，関係者で合意形成できる．

スキル不足は努力不足とも捉えられ易い側面がある．暗黙知不理解は個人の努力だけでは情報を入手することも難しく，周りの協力が必要であることを可視化できる．

暗黙知不理解に対するスキル補完は，個人の努力に加えて，組織のサポートが必要だということを可視化し，合意形成のツールとして利用できる．

新規参入者がチーム仕事の暗黙知理解を助けるツールとして利用できる．合わせて新規参入者を受け入れるチームに対し，暗黙知がどのようなリスクをもたらすかの可視化に役立つ．

5.3. Q3.共感と共創に役立つものか

共感と共創に役立つものである．

ルール違反という一見単純に見えた事故であったが，STAMP モデリングで分析した結果，ハザードが複数重なり合ったことに起因しており，その複雑性を可視化することができる．ルール違反で止まってしまうと，モラルハザードの発生や個人が悪いという判断になり易く，共感と共創とは逆の方向に行きやすい．モデリングで複雑な構造を明らかにしていくことで，モラルハザードではない構造上の複雑な問題であることを可視化できる．関係者同士の相互依存性も明らかになり，共感につながる．共感が関係者の協調を促進し，事故対策でも具体的なより良い施策を明確にすることができ，共創を促進するものである．

5.4. Q4.多能工に好影響を与えるのか

多能工の複雑性について，関係者が合意形成でき，リスクを可視化できるという点で，好影響をあたえる．

5.5. STAMP モデリングを終えて

実施後の全体的な所感を述べる．

STAMP は非常に強力なツールであることは間違いない．しかし，誰もが簡単に使えるものでは無いと感じた．実は，図 2. 初回モデリングも，自分の中では STAMP で作成したつもりであった．構造だけみればコントローラー，被コントロールプロセス，コントロールアクション，フィードバックデータに基づいた記載になっている．しかし，この段階ではハザードも相互依存関係も明らかに出来ていない．STAMP の良さは，誰が／何が悪いのかではなく，なぜ／どのようにコントロール出来ていないのかを明らかに出来ることだ．そこまで行きつくために，試行錯誤を繰り返した．実は，これが STAMP の良いところだと感じている．構造的には書けていても，そこで終わらせないところである．なぜ／どのようにコントロール出来ていないのかまで明らかにしようとすると，必然的に依存関係まで明らかにする必要があり，そこまで書かないと納得できるものが作成できないのが利点だと感じた．

単純な線形思考の場合，先に答えを仮定して，その答えに合う情報で肉付けして終わらせてしまうことも出来る．STAMP では，フィードバックプロセスがあるので，それが難しいと感じた．仮定がある側面だけから導き出されたものだとすると，フィードバックサイクルの中で矛盾がきちんと出てきてくれる．そこにハザードが潜んでいるケースが多かった．それらを紐解くと新たな矛盾が出る．それらを繰り返すと，全体がつながるようになった．

暗黙知分解の箇所は，当初を経験していたメンバーの存在も大きいと言える．STAMP の持つコントローラー，被コントロールプロセス，コントロールアクション，フィードバックデータモデリングが思考と理解を促進したのは事実であるが，経験メンバーが不在であれば，詳細の可視化までは難しかったかもしれない．

6. おわりに

システム理論に基づく事故モデル STAMP/STPA の手法が，現場の複雑な構造を可視化出来ることが分かった．暗黙知の分解とスキル不足補完といった複雑な問題にも役立つ結果が得られた．共感と共創をつくり，多能工を促進するツールの一つであると言える．多能工推進は人材不足解消に結びつくことを考慮すると，現場の人材不足解消を促すものとも言える．

効率性を追求する組織では，組織の暗黙知が大きくなり，新規参入者のハードルを上げ，同時に既存チームの負荷も上げてしまう．これらの問題を解決するのに，STAMP モデリングが有益なツールであることが分かった．

引き続き，暗黙知の構造分析とともに継承に役立て，組織における構造の問題，そこに潜むリスク可視化，問題発生時の真因特定に役立てていきたい．

プロジェクトマネジメントやプログラムマネジメントといった管理分野でも，構造理解に利用できると考えており，積極的に活用したいと考えている．

参考文献

[1] はじめての STAMP/STPA ～IoT 時代の新しい安全性解析手法～, 石井正悟
https://www.ipa.go.jp/files/000053857.pdf , アクセス日時 2018 年 3 月 10 日 10:00

[2] 知識創造企業, 野中郁次郎・竹内弘高, 訳者:梅本勝博訳, 東洋経済新報社,
ISBN978-4-492-52081-9

[3] SECI model of knowledge dimensions,
https://en.wikipedia.org/wiki/SECI_model_of_knowledge_dimensions , アクセス日時 2018 年 3 月 12 日 19:00

[4] 7S,
https://mba.globis.ac.jp/about_mba/glossary/detail-12513.html , アクセス日時 2018 年 3 月 12 日 19:00

[5] システムモデリングの新潮流:STAMP/STPA システム理論に基づく新しい安全解析手法,
日下部茂,
https://www.ipa.go.jp/files/000053967.pdf , アクセス日時 2018 年 3 月 9 日 20:00

[6] 宇宙機における不具合分析手法 CAST の適用,
寺田庸弘・星野伸行,
https://www.ipa.go.jp/files/000036240.pdf , アクセス日時 2018 年 5 月 12 日 19:00