GSNを用いた派生開発のための リバースエンジニアリングプロセス

井口 一騎[1]　中西 恒夫[2]　久住 憲嗣[3]

[1]日産自動車株式会社
[2]福岡大学工学部電子情報工学科
[3]芝浦工業大学システム理工学部電子情報システム学科

派生開発カンファレンス2023
2023年5月26日

発表内容

- 派生開発における問題点
- リバースエンジニアリング管理手法の提案
  - 想定要求仕様の記述
  - 既存ソフトウェアに関する仮説とその証明
  - 想定要求仕様の修正
- ケーススタディ
- 関連研究
- まとめ

お断り： 本発表内容は下記の既発表の原稿に基づくものです。

井口 一騎，中西 恒夫，久住 憲嗣，「派生開発のためのリバースエンジニアリング管理手法」，情処研報：ソフトウェア工学，Vol. 2021-SE-208，No. 18，8 pages，2021 年 7 月.

背景： 派生製品開発における問題点

組込みの開発現場ではソフトウェアをスクラッチから作成することは稀。

派生製品開発の方法論

- Clone-and-Own： よく似た製品のソフトウェアをコピー＆修正，後になるほど個別製品開発のコストが増大。
- ソフトウェアプロダクトライン： 全体理解志向，導入障壁大。
- XDDP： 部分理解容認，変更駆動，導入障壁はSPL より小。

背景： 派生製品開発における問題点

XDDPは本当に軽いのか？

- SPLでもXDDPでも既存ソフトウェアの現状調査は大変。
  - 既存製品のドキュメントの品質が悪い。
  - 派生元となる既存製品の要求からコードへ至るまでのドキュメントが追跡可能なかたちになっていない。
  - ドキュメントに記された仕様や設計の意図が読み取れない。
- リバースエンジニアリング（スペックアウト）の工数をどれだけ抑えられるかが勝負。
  - リバースエンジニアリングは開発者の経験や勘に頼る私的な開発活動とされがち。
  - 無駄に深くて広いリバースエンジニアリングをして工数の増大を招きがち。

派生開発プロセスXDDP（1）

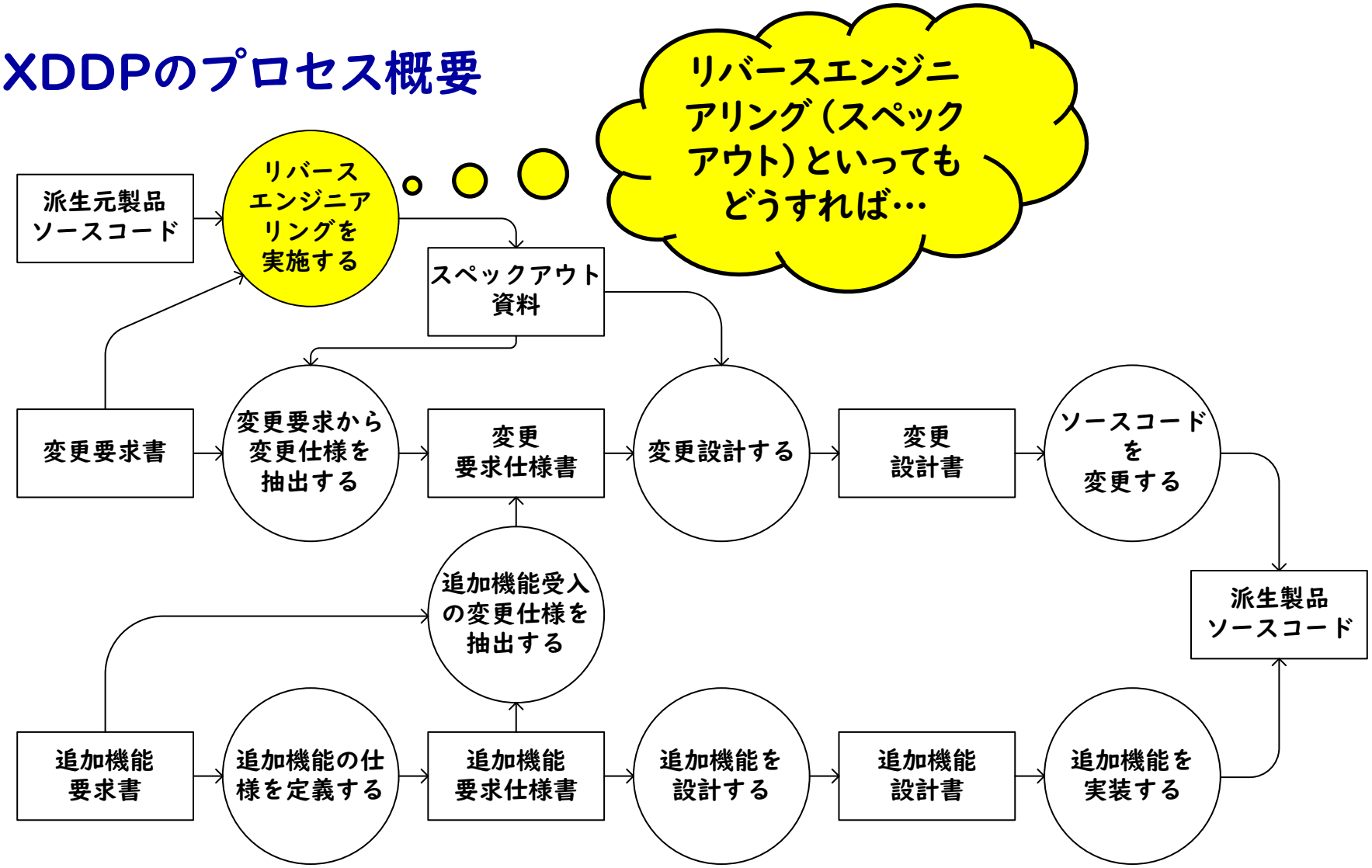
XDDP：eXtreme Derivative Development Process

- 清水吉男氏による派生製品開発のプロセス。産業界での実製品開発での実践例が多数存在する。（清水，2007）
- 派生製品開発のための既存ソフトウェアへの改修を大きく機能追加と変更に分離して捉え，それぞれのための工程を定義している。
- プロセス中，もっとも重要な役割を担うドキュメントは変更要求仕様書である。
- 変更要求仕様書に添付するかたちでトレーサビリティマトリックスも記述される。
- 既存ソフトウェアについて不明確なことはリバースエンジニアリングで明らかにする。

リバースエンジニアリングに関する具体的な方法論やプロセスは与えられていない。

派生開発プロセスXDDP (2)

XDDPのプロセス概要



リバースエンジニアリング管理手法の提案

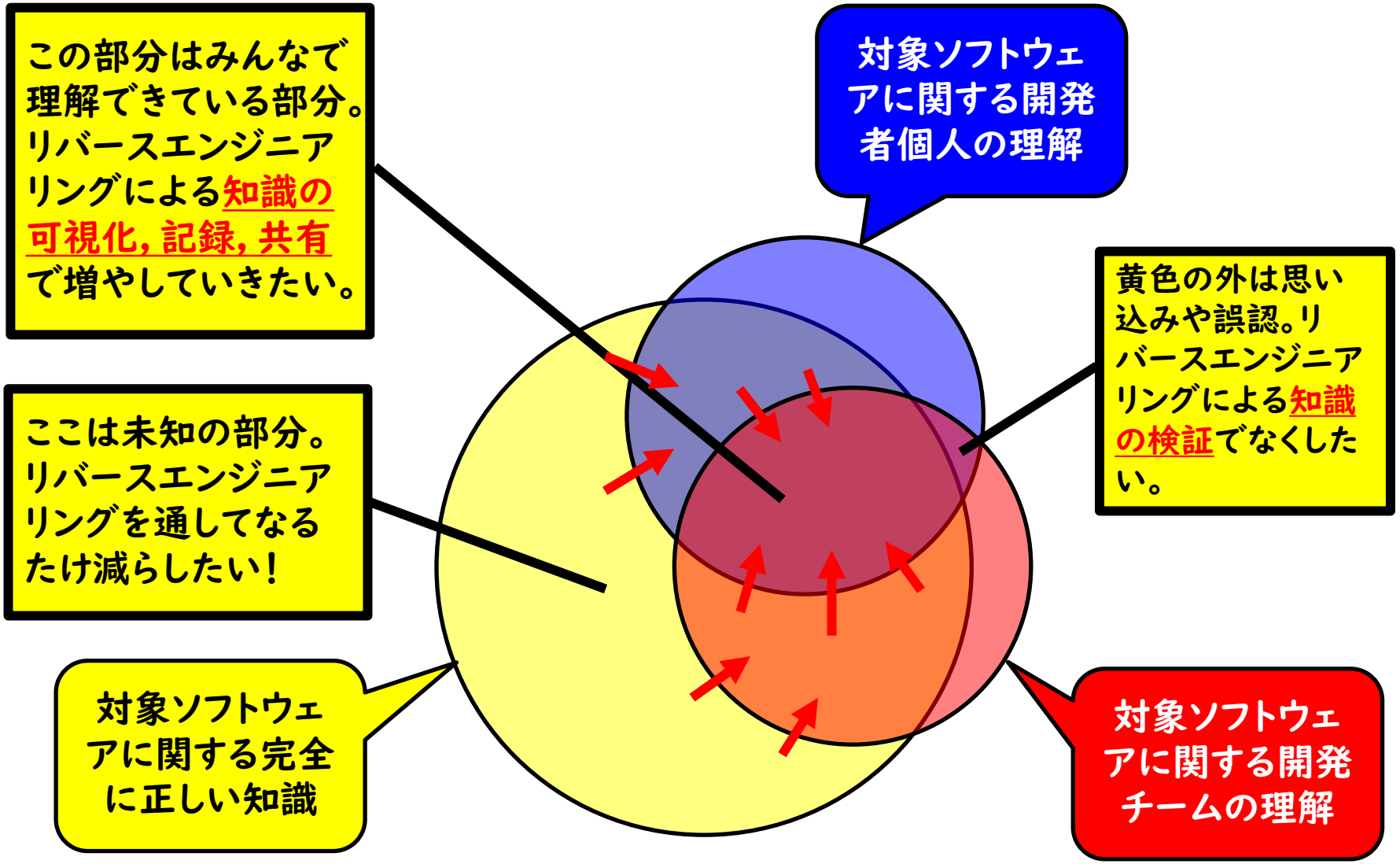
ねらい

- これまで個人的な開発活動とされがちだったリバースエンジニアリングをチームによって計画，分担実施，レビューできる活動にする。
- 目的のはっきりしない，あるいは目的を忘れた深いリバースエンジニアリングを避ける。
- 私的に，あるいは暗黙的に知られていた知識をドキュメント化し，あるいは誤解や既存ドキュメントの誤りを正す。

戦術

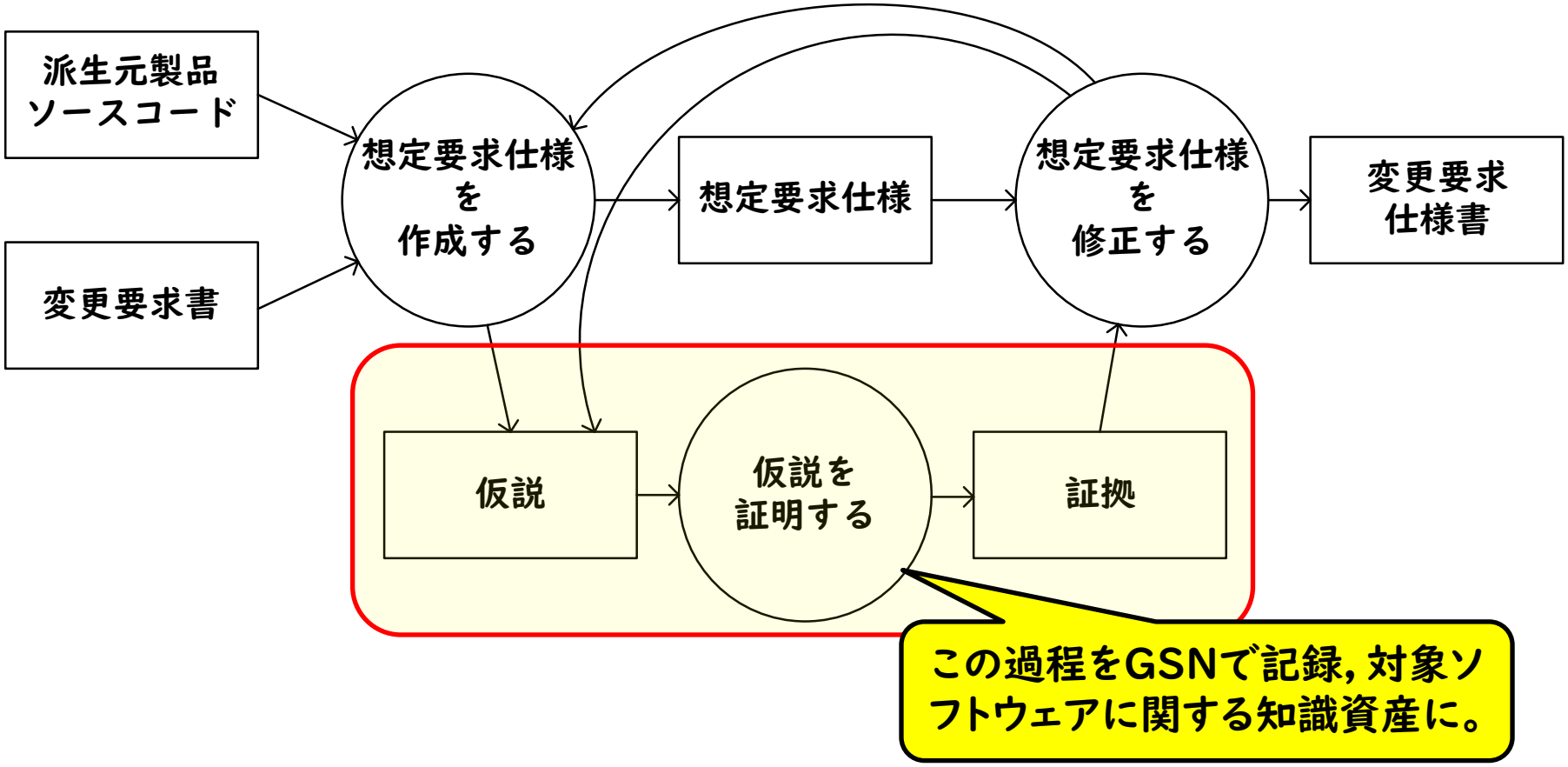
- リバースエンジニアリングのプロセス定義
- リバースエンジニアリングの目的，計画，過程，結果の記述と見える化
- それによるリバースエンジニアリングの俯瞰
- それによるリバースエンジニアリングの成果の資産化

ソフトウェア理解モデル



リバースエンジニアリングプロセス

【入力】派生元製品のソースコードやドキュメント，変更要求
【出力】変更要求仕様書



想定変更要求仕様の記述

想定変更要求仕様： ソフトウェアの仕様はおそらくこうなっているだろうとの想定や思い込みに基づいて作成した変更要求仕様書。

- リバースエンジニアリングを終えてから／やりながら変更要求仕様を導出することは行わない。（時間がかかるため。）
- 継続的に開発を行っている現場で既存ソフトウェアについて全くわからないということはあり得ない。（ただ確証がとれないだけ。）
- 明らかになっていない既存ソフトウェアの仕様記述については不確実さや曖昧さを許す。

既存ソフトウェアに関する仮説とその証明（1）

仮説： 想定変更要求仕様を作成する際に前提とした裏付けのない想定。想定変更要求仕様を作成する過程において「仮説」を洗い出す。

- 既存ソフトウェアについておそらくこうなっているのだろうと想定した事柄。
- これがわかれば既存ソフトウェアのつくりが想定できるようになると思われる事柄。

開発チームで想定変更要求仕様と仮説のレビューを行うことにより，既存ソフトウェアに関する開発チームの知識と開発者個人の理解を一致させる。

既存ソフトウェアに関する仮説とその証明 (2)

既存ソフトウェアについて立てた仮説を証明する活動としてリバースエンジニアリングを位置づける。

- 漫然とリバースエンジニアリングをしない。
- 仮説の立証に寄与しない知識獲得を制限し，リバースエンジアリングの工数の増大を防ぐ。
- 仮説をリバースエンジニアリング工程を律するハンドルとして用いる。

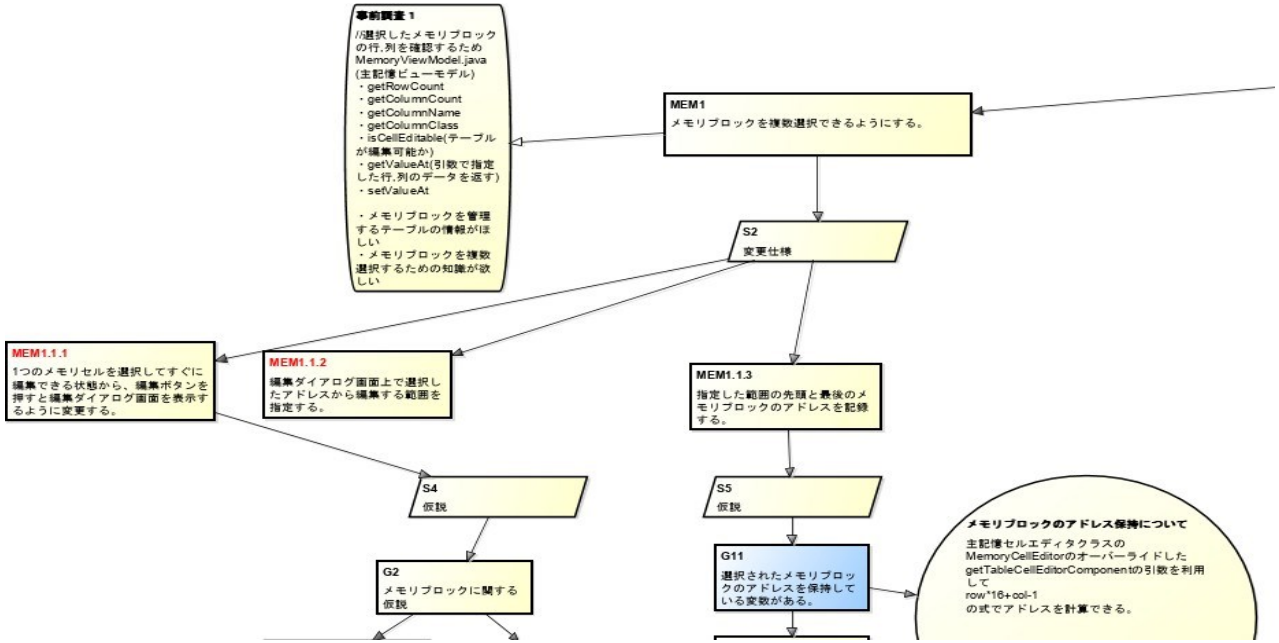
リバースエンジニアリングの過程で想定変更要求仕様の確定や修正にさらなる知識獲得が必要になった場合：

- いきなりそのためのリバースエンジアリングをしない。
- それに関する仮説を改めて立ててからリバースエンジアリングを実施する。

既存ソフトウェアに関する仮説とその証明 (3)

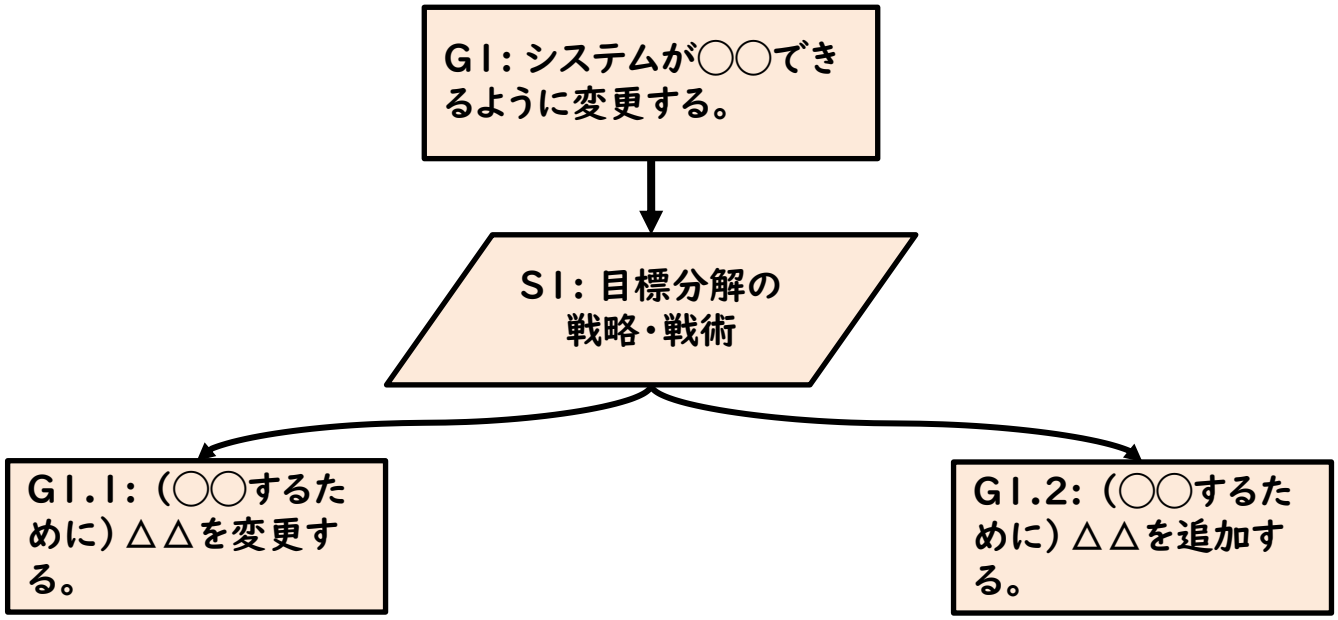
仮説の分解・証明過程のGSNによる記述

- リバースエンジニアリングの目的, 計画, 過程, 結果を開発チーム に対して見える化する。
- 変更要求仕様書には現れず死蔵されがちだった既存ソフトウェアに関する知識を資産化する。



GSN: Goal Structuring Notation (1)

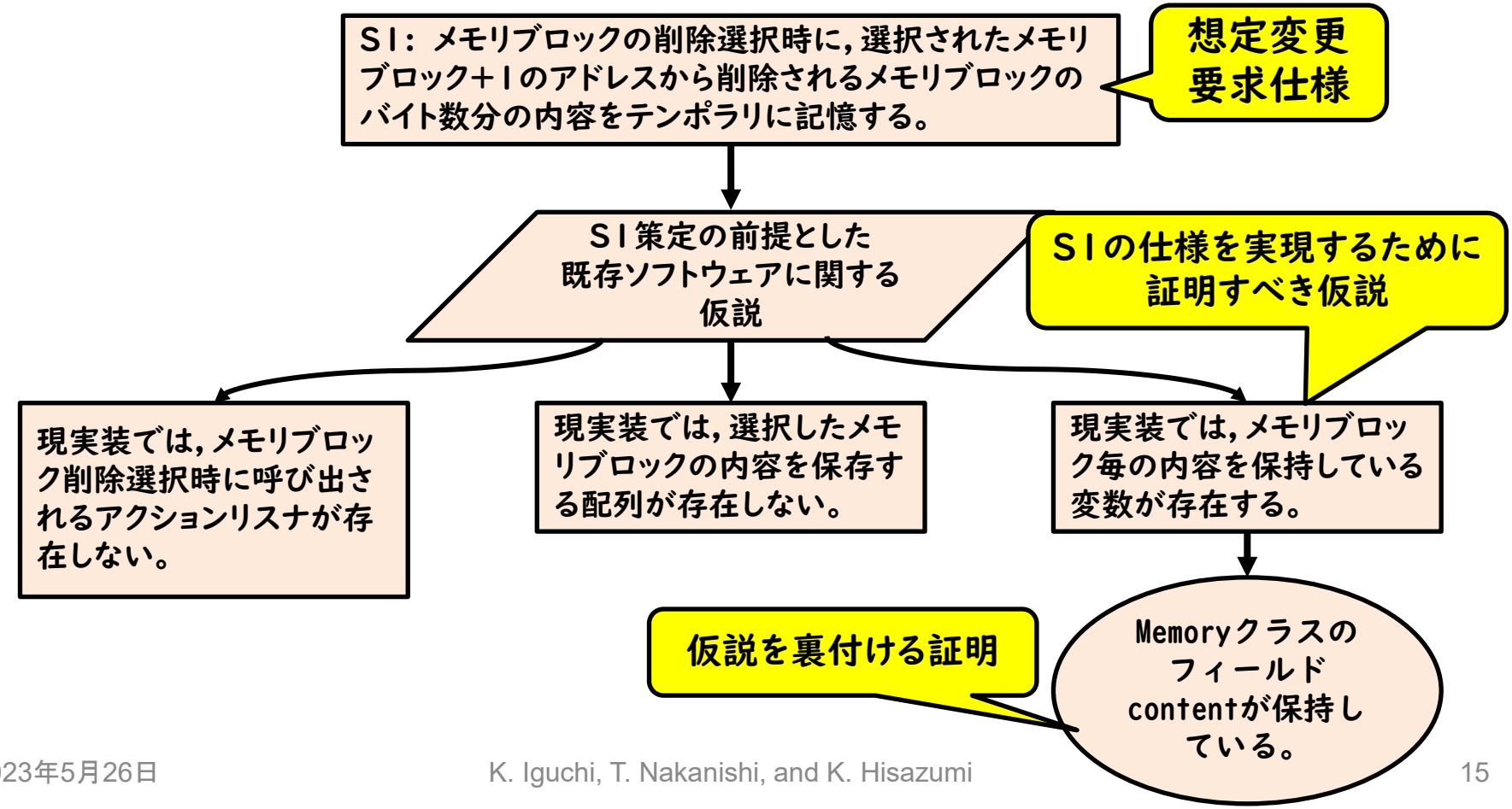
GSN: 何かしらの目標について，その目標を実現するための戦略・戦術，前提，下位目標を再帰的に書き出し，実現可能性の根拠を残す図法。



システム変更の方策（＝要求の分解と仕様化）についてはUSDMでほぼ同等の情報が記述できるので，わざわざGSNで書く意義は薄い。

GSN: Goal Structuring Notation (2)

既存ソフトウェアについて立てた仮説を証明する方策をGSNで記述する。



想定要求仕様の修正

仮説の証明結果に基づく想定要求仕様の修正

- 仮説を証明する工程で獲得された既存ソフトウェアに関する知識に基づいて想定変更要求仕様を修正する。
- 修正が生じなければリバースエンジニアリングは完了し，想定変更要求仕様は変更要求仕様として確定する。

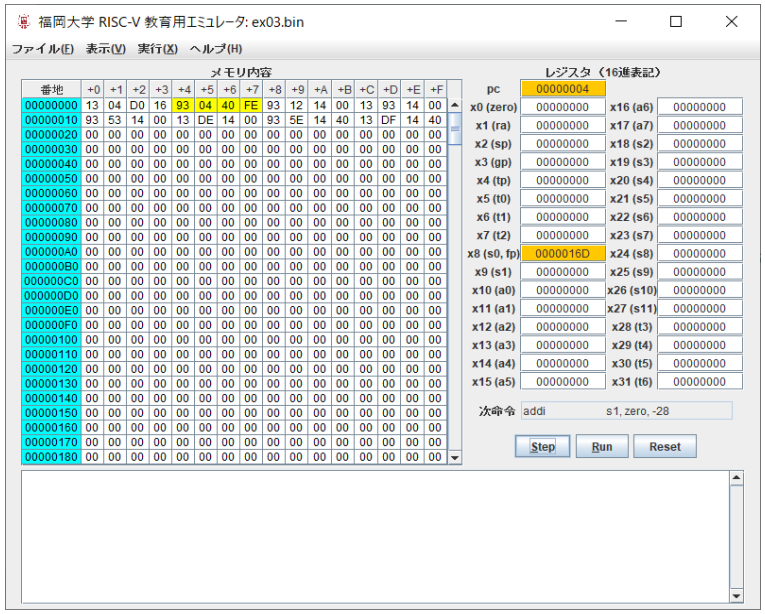
リバースエンジニアリングの終了条件

- 修正した想定変更要求仕様により新たに仮説が追加されなかったとき。
- 仮説の証明結果に想定変更要求仕様に対する反証が含まれていなかったとき。

ケーススタディの概要（1）

題材： RISC-Vエミュレータ FuRISCV

- RISC-V の基本整数命令セット（RV32I）のエミュレータ
- レジスタと主記憶の内容をGUI 上でインタラクティブに参照，編集可能
- 福岡大学工学部電子情報工学科の講義や演習で使用
- Java とSwing で実装（38 クラス 3,441 行）



ケーススタディの概要 (2)

被験者

- 被験者は 2 名。
  - 被験者A: 情報工学を専攻する学部3年生（当時）であり，学科においてすでにソフトウェア工学とJavaの講義，ならびウォータフォー ル型プロセスに基づくソフトウェア開発演習を履修している。
  - 被験者B: 工学の学位を有する30代の情報系の教員であり，ソフト ウェア工学とJavaの知識と教育経験を有する。
- 両名ともFuRISCVを演習や講義で使用した経験を有し，FuRISCVの使用方法についてはよく知っているが，その実装については全く知らない。
- 被験者Aはアルバイトとして雇用した。

ケーススタディの概要 (3)

ケーススタディの要領

- 両被験者それぞれにFuRISCVのXDDPによる派生開発を依頼する。
- 両被験者に開発プロセスや各種開発資料の作成等に関する助言は与える。
- 両被験者にFuRISCVに関する情報は一切与えない。
- 被験者Aは提案リバースエンジニアリング管理手法を用いてリバースエンジニアリングを実施する。
- 被験者Bは無手勝流でリバースエンジニアリングを実施する。
- コーディングは全工程の成果物の作成が終了した時点で行うように制限し，前工程への手戻りが発生する場合はその理由を報告するように求める。

ケーススタディの概要 (4)

派生開発要求： 隣接する複数のメモリの番地を選択できるようにし，選択された範囲のメモリに対して逆アセンブル，挿入，削除をできるようにしたい。

下位派生開発要求

- メモリブロックを複数選択し，操作を選択できるようにする。
- 選択されたメモリブロックに対して逆アセンブル，挿入，削除の操作を選択できるようにする。
- 逆アセンブルが選択された場合，選択された範囲のメモリブロックの内容に対応するアセンブリ言語命令文をコンソールに表示する。
- 挿入が選択された場合，選択された箇所以降のメモリブロックを挿入されたメモリブロックのバイト数だけ下にずらし，記憶可能なメモリアドレス数をはみ出した分は削除する。
- 削除が選択された場合，選択された箇所を削除し，選択された箇所以降のメモリブロックを削除されたメモリブロックのバイト数だけ上にずらし，メモリブロックの末尾に0で埋められたメモリブロックを削除されたメモリブロックのバイト数分挿入する。

ケーススタディの結果 (1)

- 両被験者が作成した変更要求書，変更要求仕様書（トレーサビリティマトリクスつき），変更設計書に大きな差は見られなかった。
- 被験者Bの変更要求仕様書には欠陥が見つかり，コーディング工程からの手戻りが発生した。
- 多くの工程において被験者Aは被験者Bよりも多くの工数を費やしており，全体でも550分多くの工数を費やしている。
- 手戻りとバグ修正については被験者Aは被験者Bよりも圧倒的に少ない工数で済んでいる。
- 提案手法によってソフトウェアの複雑度が大きく改善されたり，悪化したりはしなかった。

【参考】

- 被験者A：　提案リバースエンジニアリング管理手法を使用
- 被験者B：　無手勝流のリバースエンジニアリング

ケーススタディの結果 (2)

工数比較

工程	被験者A (提案手法)	被験者B (無手勝流)
変更要求仕様書作成	6h05 (27%)	2h26 (18%)
リバースエンジニアリング	8h00 (35%)	5h44 (42%)
変更設計書作成	6h00 (26%)	0h40 (5%)
コーディング	2h30 (11%)	2h20 (17%)
手戻り	0h00 (0%)	0h55 (7%)
バグ修正	0h10 (1%)	1h30 (11%)
合計	22h45 (100%)	13h35 (100%)

【参考】

- 被験者A： 提案リバースエンジニアリング管理手法を使用
- 被験者B： 無手勝流のリバースエンジニアリング

ケーススタディの結果 (3)

ソースコードメトリクスの比較

メトリクス	オリジナル	被験者A (提案手法)	被験者B (無手勝流)
クラス数	38	38	40
コード行数	3,441	3,793	3,645
平均メソッド数／クラス	3.55	3.63	3.70
平均複雑度	2.39	2.49	2.35
最大複雑度	22	22	22
平均ネスト	1.82	1.89	1.94
最大ネスト	9	9	9

【参考】

- 被験者A： 提案リバースエンジニアリング管理手法を使用
- 被験者B： 無手勝流のリバースエンジニアリング

関連研究（1）

XDDPのスペックアウト工程におけるリバースエンジニアリングの未成熟はこれまでも指摘されている。

- 変更要求仕様の不正確さに起因する不具合と工数の増加
- 不十分な既存ソフトウェアの調査が変更要求仕様の不正確さの主因
- 既存ソフトウェアを俯瞰した合理的な調査プロセスの必要性

関連研究 (2)

中井，2010

- 無知見プロジェクトにおける既存ソフトウェアの調査プロセスを定義
- 既存ソフトウェアの設計書を入力とし，処理フローを抽出してUSDMで記述し，それをもって既存ソフトウェアを俯瞰
- 必要な箇所のみ詳細な調査を実施して変更要求仕様書を作成

飯泉，2015

- 既存ソフトウェアの調査を事前調査，変更箇所調査，影響範囲調査の3ステージに分離
- それぞれの調査の目的を明確化しプロセスを定義
- 事前調査により既存ソフトウェアを俯瞰

関連研究 (3)

関連研究と本研究の共通点

- リバースエンジニアリングの実施にあたり既存ソフトウェアの俯瞰に重きを置く点

関連研究と本研究の相違点

- 関連研究は品質の高い変更要求仕様書の作成を目的とする。
- 本研究はリバースエンジニアリングの目的，計画，過程，結果の記述と形式知化を目的とする。
- 関連研究は設計書やソースコードをリバースエンジニアリングの出発点とする。
- 本研究は想定変更要求仕様をリバースエンジニアリングの出発点とする。
  - 既存ソフトウェアの現状に束縛され過ぎない派生開発ができる。
  - 既存ソフトウェア資産に明示的に書かれていない知見を検証対象に取り込める。
- 無知見プロジェクトの場合は差は出ないものと思われる。

まとめ

- 派生開発プロセスXDDP向けのリバースエンジニアリング管理手法を提案
  - 想定変更要求仕様を記述し，リバースエンジニアリングの結果に基づいてそれを漸次的に修正，最終的に変更要求仕様を完成
  - リバースエンジニアリングを既存ソフトウェアに関する「仮説」を立証する活動として位置づけ
  - GSN によりリバースエンジニアリングの目的，計画，過程，結果を記述／見える化／資産化
- ケーススタディによる評価
  - 派生開発に要する全体の工数を下げることはかなわなかった。
  - 手戻りやバグ修正に要する工数は，絶対値でも割合でも大幅に抑えられた。
  - 工数のほとんどが派生製品の開発に直接的に必要な作業に使われた。

参考文献

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年.
2. 中井 栄次，「XDDP適用による無知見プロジェクトのプロセス改善」，派生開発カンファレンス2010，2010年6月.（*https://affordd.jp/conference2010/xddp2010_P4.pdf*，最終アクセス日：2021年6月7日）
3. 飯泉 紀子，足立 久美，清水 吉男，宇田 泰子，夛田 一成，川井 めぐみ，伊藤 友一，「変更の影響範囲を特定するための『標準調査プロセス』の提案」，ソフトウェア品質管理（SQiP）研究会第30年度第6分科会Aチーム成果報告，2015年2月.