JASPICトワイライトフォーラム

NTT DATA
Global IT Innovator

IoT時代のCMMI再考
～CMMIのシステムエンジニアリング視点を見直す～

2017年11月13日

(株)NTTデータ　技術統括革新本部
端山　毅　（JASPIC理事）

IoT時代のCMMI再考
～CMMIのシステムエンジニアリング視点を見直す～

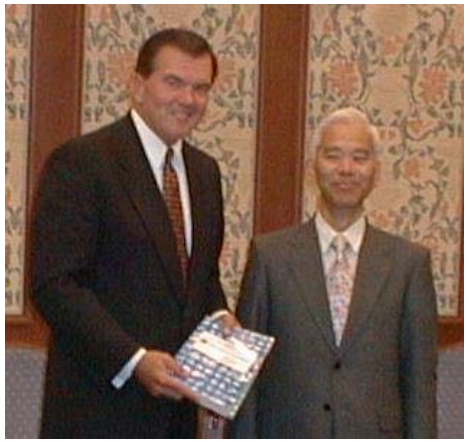
〈構成〉

■ 第1部
システムズエンジニアリングに注目すべき理由
そのとき、なぜCMMIが利用できるのか

■ 第2部
CMMIから読み取れるシステムズエンジニアリングの示唆

ペンシルベニア州知事来日に合わせ、1999年5月17日のホテルオークラ朝食会席上にて、CMM v1.1 公式日本語版の SEA Web 上での公開開始 (1999年5月10日より) についての記者発表を行ないました。また、刷り上ったばかりの製本版一式を、SEA-SPINを代表して坂本氏が、州知事とカーネギーメロン大学(CMU)理事に贈呈しました。

ソフトウェア技術者協会のホームページより
http://www.sea.jp/CMM/



左： Tom Ridge (当時ペンシルバニア州知事、国土安全保障省初代長官)
右： 坂本啓司(元ソフトウェア技術者協会代表幹事
JASPIC設立時(2000年)理事、JASPIC初代運営委員長)

坂本啓司氏は、2001年8月10日ご逝去されました。享年54才
日経コンピュータ 7月30日号に寄稿された論文
「″CMM″ で陥りがちな ″罠″ を理解せよ」が遺稿になりました。

第1部

システムズエンジニアリングに注目すべき理由
そのとき、なぜCMMIが利用できるのか

CMMIの“I”はIntegration

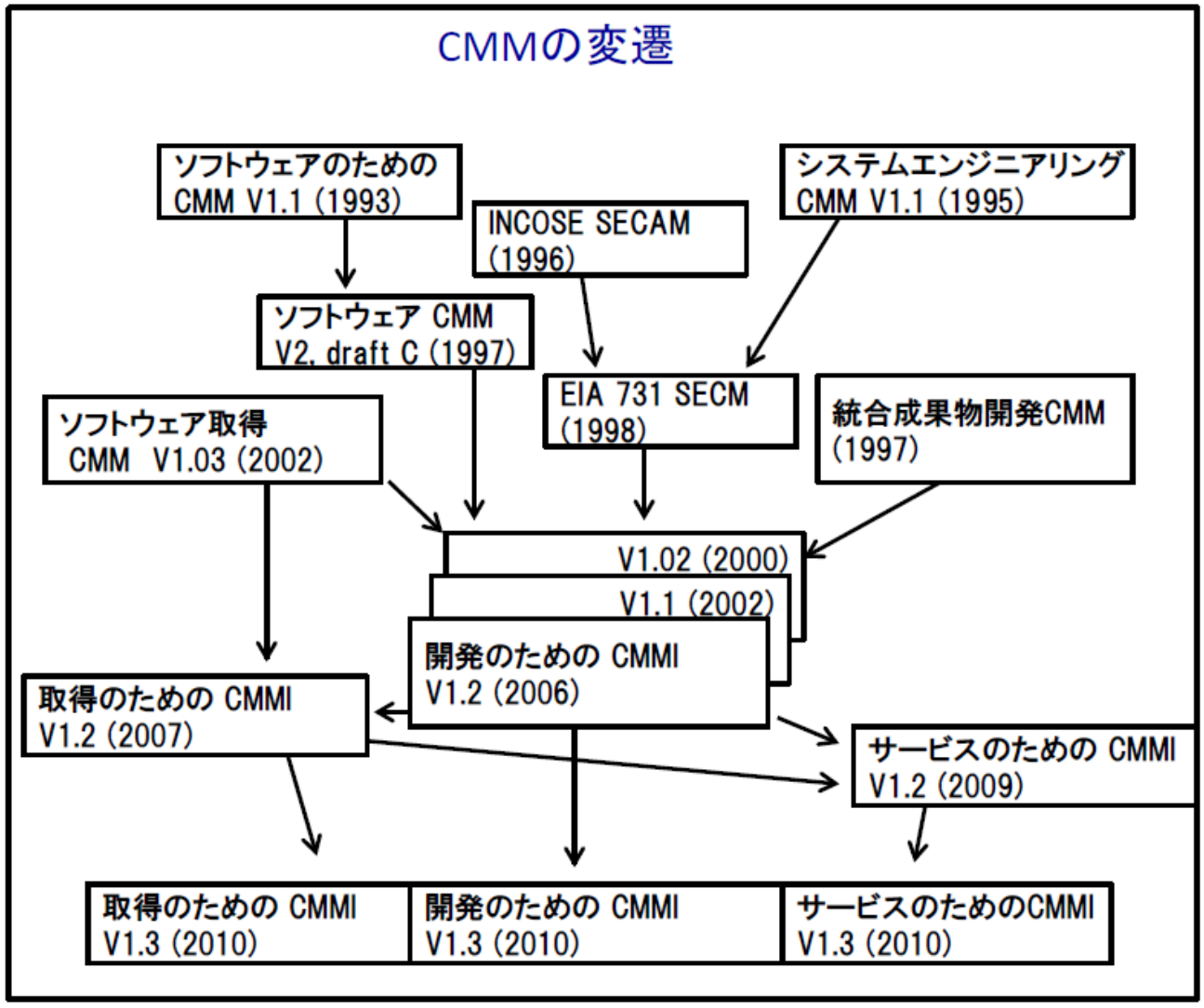


図 1.2: CMM の変遷 [6]

4つのモデルの「統合」
- Software CMM
- Software Acquisition CMM
- Systems Engineering Capability Model
- Integrated Product Development CMM

CMMIのエンジニアリングプロセスは
システムズエンジニアリングの考え方に
沿っている

要件開発
技術解
成果物統合
検証
妥当性確認

「システム」視点が求められる時代

- IoTの広がり(AI, ビックデータ、クラウド)
- 多様なセンサー、アクチュエータ、装置がつながったシステム
- 多様なアプリケーション
- 今までにないもの
- System of Systems
- 共創の時代:　多様な専門家の協業
- ジョブ理論:　真のニーズの探求

  業務プロセス、人の振る舞い、価値観

広い視野と協力関係

CMMIの中のシステムズエンジニアリングならではの記述例

技術解 SP 1.1 解の選択肢と選定基準を策定する

解の選択肢および選定基準の考慮事項:

- 開発、製造、購入、保守、および支援などの費用
- 成果物の適時性、安全性、信頼性、および保守性などの主要な品質属性要件の達成
- 成果物構成要素の複雑度および成果物関連のライフサイクルプロセスの複雑度
- 成果物の運用および使用の条件、運用形態、環境、および成果物関連のライフサイクルプロセスの変動に対する頑健性
- 成果物の拡張と発展
- 技術制限
- 構築手法および材料に対する感度
- リスク
- 要件および技術の進化
- 廃棄
- 最終利用者および運用者の能力と制約
- 「商用市販の成果物」の特性

設計の考慮事項として
製品のライフサイクルを強調

日本のCMMIコミュニティはシステムズエンジニアリングを意識しなかった

【違和感がなかった】

- SW-CMM時代の利用者が少なく、CMMIから入った人が多い
  ⇒ ソフトウェアとの違いを意識せず、そういうものだと思った
- 指導者(ＬＡ－Inst)が解釈して説明した
  ⇒ 適度に読み飛ばした
- ソフトウェア技術者にとっても、それほど違和感はなかった
  ⇒ 自社標準の解釈で対応できた
- 日本企業にはシステム視点があった
  ⇒ 顧客重視、長期関係
- ソフトウェア技術者、ハード技術者、システム技術者という区分がない
  ⇒ 必要なことは学ぶ、受け入れる
  ⇒ 一つの企業の中で立場が代わる
  ⇒ 同じ会社の中に多様な役割の人がいて、協力するのは当然
- 多くのパートナーと協力するのは自然なこと
  ⇒ 長年の取引慣行

IoT時代にシステムズエンジニアリングを見直すべき

- ISO/IEC/IEEE 15288:2015
  “Systems and software engineering -- System life cycle processes”
- INCOSE：International Council on Systems Engineering
  - EIA731 SE-CM
  - Systems Engineering Handbook (Fourth Edition)
- Guide to the Systems Engineering Body of Knowledge (SEBoK)

➢ システム工学からシステムズエンジニアリングへ
数理的な最適化問題から、ライフサイクル、多数の専門分野の連携へ発展

➢ 複雑性と不確実性に対峙してきたシステムズエンジニアリングの歴史
- ベル研発祥、DoD、NASA、航空宇宙産業
- 多様な業界へ拡大：インフラ、エネルギー、医療、自動車、....
- 機械/電気から、人間系を含むシステムに対象を拡大
- VUCA： Volatility、Uncertainty、Complexity、Ambiguity

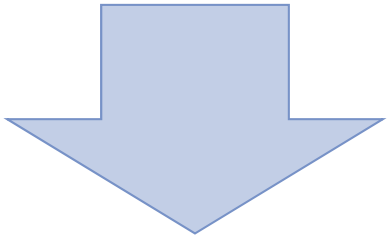
システムズエンジニアリングも時代とともに発展し続けている

システムズエンジニアリングから何を学ぶか

- ◆増大する複雑さへの挑戦
- ◆応用分野の拡大
- ◆技術と管理の融合
- ◆人的要因との格闘
- ◆グローバル化対応

ソフトウェアエンジニアリングで取り上げている課題は、システムズエンジニアリングでも同様に取り上げている

誰がやっても難しい！！



問題解決の事例と対策を学ぶ

苦労話は聞く価値がある

システムズエンジニアリングの特性

ソフトウェアエンジニアリングと比べて

1. 利害関係者が多様（ソフトウェアでも多様な場合はあるが）
   - ◆ 関与する専門家も多様
   - ◆ コミュニケーションが難しくなる
2. 構成要素の種類が多様
   - ◆ 構成要素のライフサイクルも異なることがある
   - ◆ 製品の寿命も異なる(経年劣化も考慮する)
3. 開発プロセス／管理プロセスが多様
   - ◆ 開発対象物によって開発アプローチ／標準も多様になる
   - ◆ 検証、妥当性確認の方法を独自に考える必要が生じることがある
   - ◆ テーラリングの度合いが大きくなり易い
   - ◆ 利用する装置、ツール、道具が多様になる
   - ◆ 測定尺度が多様になる
4. 適用される技法の種類が多い

情報量が多い

システムズエンジニアリング教材としてのCMMI

- システム工学の教科書は昔からある
  数理解析、最適化問題が中心
  ライフサイクルプロセスを取り上げたものは少ない
- 最近の資料は日本語が少ない(ISO/IEC15288, SE-Handbook, SE-BOK)
- 汎用的なシステムズエンジニアリングは、抽象的で分かりにくい

比較的新しい教科書
「システム工学―問題発見・解決の方法―」
「システム工学―定量的な意思決定法―」
井上雅裕他、オーム社
ライフサイクルプロセスを取り上げている

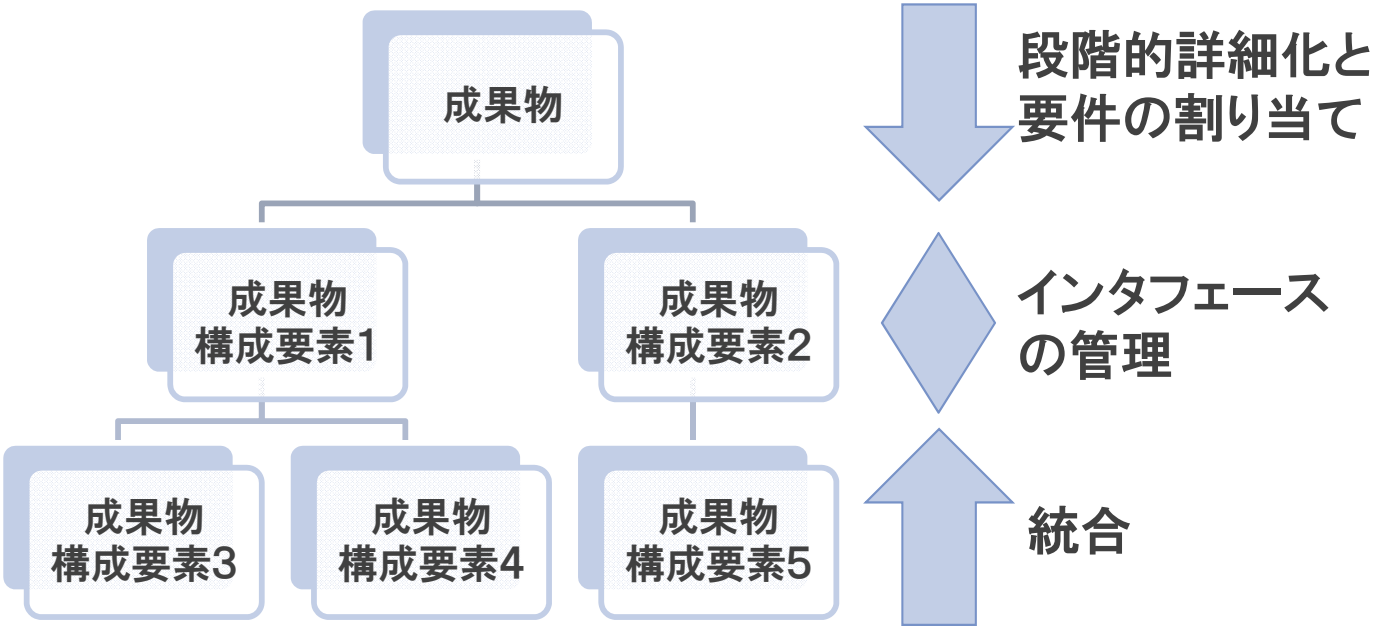
- CMMIは、
  - ソフトウェア開発だと思えば、ソフトウェア開発の話として読める
  - よく見ると、ソフトウェア開発以外の話がたくさん埋め込まれている
  - ソフトウェア開発と同じ構造に沿って読める、
    例示のコンテキストが理解し易い
  - ソフトウェアエンジニアとシステムズエンジニアの協力を意図した文書

CMMI:システムズエンジニアリングへの窓口

第２部

CMMIから読み取れる
システムズエンジニアリングの示唆

成果物の階層構造



複雑さへのアプローチの基礎

- 利害関係者のニーズ
- 顧客要件
- 成果物要件
- 成果物構成要素の要件
- 運用の考え方
- 必要とされる機能性
  および品質属性
- 導出要件
- 成果物アーキテクチャ
- 成果物統合の戦略
- 成果物統合環境

事業目標の強調

「事業目標」: CMMI-DEV V1.3での出現回数　125回

- システム開発やプロセス改善が実施される状況を説明
- 上位の目的達成を強調
- 高い視野に立って、組織としてのBenefit実現を目指す
- 目標の階層構造、ブレークダウンを解説
- 「共有ビジョン」による一体化　⇒　多様な専門家の連携手段

現実への適用を促進する実践的内容

ソフトウェアとシステムの橋渡し

上流での検討事項

- ◆ 顧客要件の優先付け
- ◆ 要件の妥当性確認
- ◆ インタフェース要件
- ◆ 「検証」、「妥当性確認」の実施に対する制約

  検証や妥当性確認の方法や実施環境の検討の早期実施

- ◆ 設計に対する科学的、工学的な検証や妥当性確認を推奨
  - シミュレーションパッケージ
  - プロトタイピングツール
  - 統計パッケージ
  - 動的システムモデリング
  - オンライン技術データベースおよび刊行物の利用
  - プロセスモデリングツール

随所でモデリングやシミュレーションを推奨

ソフトウェアエンジニアリングで、もっと多用すべきだろう

作る前に、性能、信頼性、工数などを評価して、リスクを低減する

多様な利害関係者

利害関係者の例

- 顧客
- 最終利用者
- 供給者
- 構築者
- 検査者
- 製造業者
- ロジスティックス支援要員

利害関係者 vs. 直接の利害関係者

- ニーズ
- 期待
- 制約
- インタフェース
- 運用の考え方
- 成果物の考え方

連携相手を増やして、
自らの可能性を広げる

- 考慮すべき利害関係者
- 利害関係者が生じる場面
- 利害関係者との協力方法
- 誰とどう組むか

多様な専門家の協力

顧客担当
職務機能部門
品質保証部門
マーケティング部門
技術部門
事業部門

機械工学
ロボット工学
人間工学

複合材料
人工知能
セキュリティ
安全性

インタフェースの種類
機械、流体、音響、電気、気候、電磁気、熱、メッセージ、
およびヒューマン・マシンまたはヒューマンインタフェース
電気的、機械的、光学的

IPM SP1.6 チーム群を確立し保守する
OPD SP1.7 チーム群を体系化し、編成し、そして運営するための
「組織の規則および指針」を確立し保守する。

- チーム群を体系化し編成するための「組織の規則および指針」
- チーム群のための規則および指針(チーム群の運営規則)
- チーム体系、チーム憲章
- プロジェクトの共有ビジョン
  「共有ビジョン」
  任務、目標、期待される行動、価値観、および最終的な実施結果を含むような、指針となる原理の共通の理解。

成長、能力開発の道を示唆

課題に対処するための解の選択肢

- 正式評価プロセス
  - 正式評価プロセス適用指針
  - 選択肢の評価基準
  - 解の選択肢
  - 評価手法
- 評価対象
  - 機器またはソフトウェア製品
  - アーキテクチャ
  - 設計
  - 供給者
  - エンジニアリング支援環境／ツール
  - テスト環境
  - 自製、再利用、購入
- 直接の利害関係者の複数の要求
  - 実装に対するリスク評価
  - 選定の論理的根拠の文書化と伝達

- 評価手法の例：
  - テスト
  - モデリングおよびシミュレーション
  - エンジニアリングの検討
  - 製造の検討
  - 費用の検討
  - 事業機会の検討
  - 調査
  - 現場経験およびプロトタイプに基づく推定
  - 最終利用者のレビューおよびコメント
  - 専門家または専門家のグループによってもたらされる判断（例えば、デルファイ法）

意思決定根拠の明示化

利害関係者が多様化すると一層重要

多様な成果物

構成管理下に置かれる作業成果物の例を以下に示す：

- ハードウェアおよび機器
- 図面
- 成果物仕様
- ツール構成
- コードおよびライブラリ
- コンパイラ
- テストツールおよびテストスクリプト
- 導入ログ
- 成果物データファイル
- 成果物の技術資料
- 計画
- ユーザストーリー
- 反復バックログ
- プロセス記述
- 要件
- アーキテクチャ文書および設計データ
- プロダクトライン計画、プロセス、および中核資産

例示の種類が豊富

技法

- 技術実証
- インタフェース制御の作業グループ
- 技術制御の作業グループ
- 暫定的なプロジェクトレビュー
- 最終利用者から入手した質問票、インタビュー、および（運用、維持、および開発面の）シナリオ
- 運用、維持、および開発面のウォークスルー、および最終利用者のタスク分析
- 利害関係者と共に品質属性を引き出す研究集会
- プロトタイプおよびモデル
- ブレーンストーミング
- 『品質機能展開』
- 市場調査
- ベータテスト
- 文書、標準、または仕様などの情報源からの抽出
- 既存の成果物、環境、およびワークフローパターンの観察
- ユースケース
- ユーザストーリー
- 成果物機能性の小さく漸進的な『階層縦断的なまとまり』を引き渡すこと
- 投資対効果の分析
- リバースエンジニアリング（過去から引き継がれた成果物向け）
- 顧客満足度調査

例示の種類が豊富

ソフトウェアエンジニアリングへの示唆

- 上流で、システム全体、業務全体、運用環境などを見渡して、リスクを洗い出す
  - リスクを低減するための分析評価に注力
    - ソフトでは、プログラムを書いて分析できる
      - ⇒ 最適化、チューンが場当たり的ではないか
        数理モデルを作って解析するという姿勢が弱い
      - （昔はクロック数えたのでしょうが）
    - 業務も含めて、最適化を追究すべきではないか
      - ⇒ 人間相手だから難しいと諦めていないか
  - プロトタイピングやシミュレーションを多用すべきではないか
  - 実験室⇒実証プラント⇒実用化　のようなステップも要る

多くのエンジニアの共通言語としてのシステムズエンジニアリング

- システムズエンジニア専業は少ない
- 元々は、機械、電気、電子、化学、土木、建築など、専門分野を持つ
- 顧客/市場ニーズに対応して「システム」作る必要が生じる
- システムズエンジニアに成る
- 他の領域のエンジニアと協力するための基盤
- ソフトウェアエンジニアも、システムズエンジニアに成る
  - 龍や馬に成る

- 業務システムでも、組込みシステムでも、全体を見渡している技術者や管理者は、多分、システムズエンジニアリングを駆使している
  - 叩き上げか、教育されたかの違い
- 迅速さを求められる時代
  - 教育も必要
  - 学習した上で、実践を通じて、独自の境地を切り開く

まとめ

現在のビジネス環境において、多くの人たちが類似の問題に直面している
VUCA： Volatility、Uncertainty、Complexity、Ambiguity

システムズエンジニアリングもソフトウェアエンジニアリングと類似の課題と向き合っている
そのアプローチには、学ぶべきところがある

> 【私見】:ソフトウェアエンジニアリングでは、標準、ライフサイクルプロセス、方法論、成果物、プロジェクト管理様式、メトリクス、見積り手順などが、比較的整備されているので、システムズエンジニアより、自ら考える機会が少なく、応用力が弱い可能性がある。

CMMIはその入門書としてお手軽（日本語、無料）

再読を勧めます

NTT DATA

Global IT Innovator