4A-06

システムズエンジニアリングの効果検証：学生ロケットプロジェクトへの導入実験

伊藤夏青（横浜国大），秋元賢介（千葉工大），小布施聡（慶大）菅田徹也（横浜国大），
玉越大資（首都大），吉本直哉（横浜国大），嶋津恵子（産業技術大）

Observation of Systems Engineering Effectiveness -A Hands-on Practice: A Small Size Hybrid Rocket Mission-

Kao Ito (Yokohama National University), Kensuke Akimoto (Chiba Institute of Technology),
Satoshi Obuse (Keio University), Tetsuya Sugata (Yokohama National University),
Daisuke Tamakoshi (Tokyo Metropolitan University), Naoya Yoshimoto (Yokohama National University,
Keiko Shimazu (Advanced Institute of Industrial Technology)

Abstract

Japanese industries place more expectations on the space area. In this paper, we report that the result of employing systems engineering methodology to our Small Size Hybrid Rocket Mission. We recognized the effectiveness of engineering concurrently and regarding system interfaces as serious, those are principle of international standard of Systems Engineering, ISO/IEC 15288.

1. はじめに

現在，宇宙産業はグローバル規模で成長している産業であり，今後もその成長が見込まれている．しかし日本は宇宙産業における競争力では，欧米に大きく水をあけられており，産業に必要な工業力を早期に成熟させる必要がある[1]．一方，日本でも宇宙産業のすそ野は広がりつつあり，関東圏の大学生が大学を横断して設立した小型ロケット製作サークル CORE(Challengers of Rocket Engineering)も，その一つである[2]．CORE は宇宙工学以外の工学を学ぶ学生らが，必要な知識を習得し，小型ロケットを企画・設計・製作し，打ち上げまでを実体験することを目的にしている．習得する領域は，空気力学，燃焼工学，機械工学，制御工学，電気電子工学等専門性の高い分野と，プロジェクトマネジメントである．2007 年の設立から現在までに，約 20 機のロケットを打上げている． 本書では，これまでの記録からミッション達成に至らなかったケースを取り上げ，前述した専門的分野の利用に加え，横断的工学であるシステムズエンジニアリングを適用した場合の改善状況を考察した．

本書の題材は CORE が過去に行った「Helix プロジェクト」である．このプロジェクトでは発射されたロケットを指定した位置に着地させることをミッションとし，2014 年 11 月にプロジェクトを開始し，2015 年 3 月に打上実験を行った．

2. Helix プロジェクトでのミッション失敗

このプロジェクトでは，ミッション達成のための方策としてパラフォイルを利用した制御機構を用いた．より具体的には，ロケット本体にこれを搭載した状態で上空に発射し，噴射停止後，最高到達地点である上空 300 メートル付近でこれを放出する．その後，ロケット本体に搭載された位置情報獲得用センターの値を利用し，組み込んだソフトウエアプログラムで地上目的地まで自律飛行する．

実際の打上げ実験では，パラフォイル放出直後にパラフォイルとロケット本体を接続した複数の紐が互いに絡まった．これにより，パラフォイルが閉じたままとなった．予定していた制御は行われず，ロケット本体は自然落下状態となり地上に墜落した．

プロジェクト修了後，我々は，プロジェクトの進め方をレビューした．その結果，ロケット本体とパラフォイル機構をそれぞれ滝型モデルで開発し，それらが終了した時点で統合するという方法を採用していたことを確認した（図 1）．

前述の紐同士の絡まりは，直感的には図 1 の「System Integration and Testing」のプロセスを充実させることで，発生を防ぐことができるように見える．一方，ロケット打上ミッションは，打上げ後の状態を模擬した事前試験が困難な場合が多い．本プロジェクトでも，前述のプロセスではロケット本体にパラフォイル機構を納め，蓋が(発射に耐えられる程度に)閉じることを確認するに留まっている．

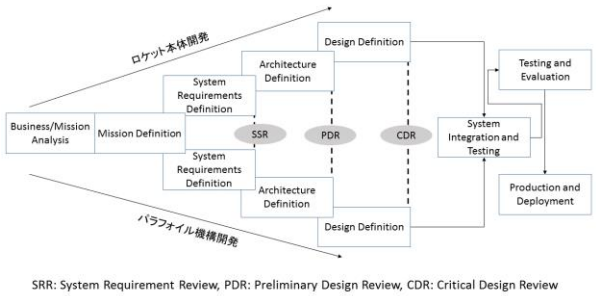

図 1 Helix プロジェクト：2 つの滝型開発と統合

3. システムズエンジニアリング外観

INCOSE (The International Council on Systems Engineering)が提唱する ISO/IEC 12588 のシステムズ

エンジニアリング・ライフサイクル・プロセス標準は，対象とする問題をシステムの構築と導入によって解決することを遺漏なく進めることがその狙いである[3]．これを実現するために，ライフサイクルの初期から終焉まで，（部分最適化だけでなく）全体最適化を意識して工学的作業を行うよう，コンカレントエンジニアリングの実践を前提としている(図 2)．視点や粒度の異なる作業プロセス群を同時に実施しながら，それらの最適な関係性を保つ（全体最適化の実現）ことで，妥当性の高いシステム設計と実装を実現することを目指している．

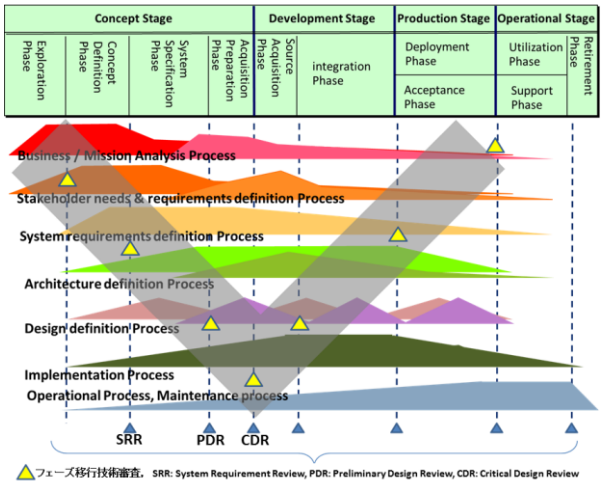

図 2 システムズエンジニアリング：V モデル型の開発

4. システムズエンジニアリング導入の影響

我々は Helix プロジェクトをシステムズエンジニアリング型で再実施中である．具体的には，ロケット本体の開発とパラフォイル機構を並行して実施し，さらに 2 つの内部の開発プロセスも並行して行う．そして，図 2 に示すように，フェーズ移行時にシステム全体としての設計上の最適性を確認するための移行審査を技術評価の観点で行う．この作業が，3 章で整理した，失敗したプロジェクトで採用した開発方法と大きく異なる点であった．

設計作業開始後の最初のフェーズ移行審査は，図 2 の Concept Definition Phase から System Specification Phase に移行する段階(SSR: System Requirement Review)である．審査に要した設計図は，3 つ，つまり，ロケット本体の設計図とパラフォイルの設計図と，さらにこれらの結合仕様となる設計図である(図 3)．これに対し，Helix プロジェクトのこの段階は，図 2 の SSR が示すように，ロケット本体とパラフォイル機構それぞれ個別に行っている．したがって，審査対象もこれらの 2 つの設計図だけが用いられた(図 4)．

本書では紙面の都合上，設計図ではなく外観図を用いた．特に図 3 に示した設計図を参照すると，結合に関するものが増えているだけでなく，その仕様の影響を受け，機体本体の設計図に変化が発生しているのがわかる．つまり，システムズエンジニアリングを導入したことで，より早期に，この 2 つのシステム・要素の結合に関し，技術的な対応がとられたと言える．

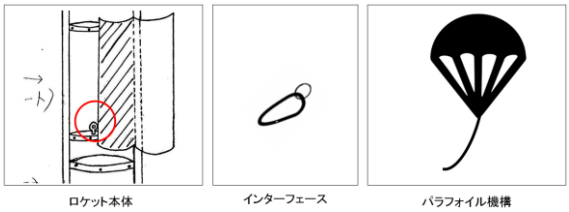

図 3 SE 導入後の SSR における 3 つの設計図

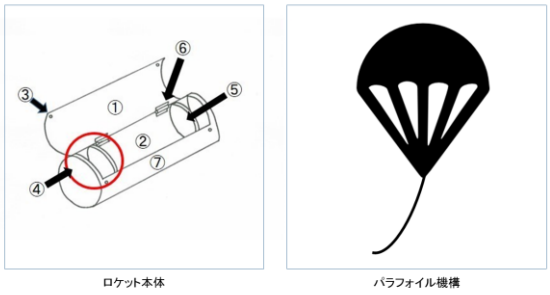

図 4 SE 導入前での SSR における 2 つの設計図

5. 結論と考察

本書では，小型ロケット打上げプロジェクトの代表的な失敗例を取り上げ，システムズエンジニアリングの導入の効果を観察した．その結果，決定的となる統合技術確認を，プロジェクトの初期の段階に，シフトさせることが可能になると分かった．同プロジェクトでは，打上直前の最終統合作業で，パラフォイルの紐を通す穴のロケット本体への施工が未実施だったことも発見されている．現場にあった工具を利用し，至急対応を行っているが，これも統合箇所の検証を後回しにしたことが原因だと考えられる．

歴史と経験からノウハウを継承することは重要であるが，この方法だけでは，未知の失敗に事前に対応することは難しい．システムズエンジニアリングのフレークワークを導入することで，今後 CORE の後輩たちが，より高度なミッションに到達できるよう助言していきたい．

参考文献

[1] FORTRON'S 2014 SPACE COMPETITIVENSS INDEX, A Comparative Analysis of How Countries Invest In and Benefit from Space Industry, Fortron Corporation,2014

[2] http://corerocket.lolipop.jp/, 2016 年 1 月 7 日参照

[3] INTERNATIONAL STNADARD, ISO/IEC/15288, System and Software Engineering – Life Cycle Processes, 2008, 2015 改訂