青山学院バーチャル・キャンパス基盤システムの構築

LUCINA-based Development of "Infrastructure System for Virtual Campus at Aoyama Gakuin University"

羽田昭裕

要　約　青山学院が推進するAMLプロジェクトは，社会・経済構造の変化に対応した新しい教育の方法と基盤の提供を目的としている．日本ユニシスはこの基盤システムを担当し，開発技法LUCINAを採用して開発した．このプロジェクトの特性は次の二つである．まず，多様な環境に対応する柔軟性を持たせるため，基盤システムのデータと通信の技術としてXMLとCORBAを採用した．そして，協調的な演習を中心とすることから，授業はやり直すことができず，利用シナリオを完全には事前定義できない．前者の特性は技術的なリスクやスキル・リスクを，後者は要求に関するリスクをもたらす．

アーキテクチャ中心の開発プロセスは，新しい技術や変化していく要件に応える構造を持ち，リスク管理を軸として品質と納期を守るプロジェクト管理に貢献する．LUCINAは，このような開発プロセスの一つである．

要求，技術，スキルのリスクをどうのように軽減していったかを概観する．このプロジェクトはLUCINA開発プロセスの効果を示す事例となった．

Abstract　To provide new educational method and infrastructure which accords with nowadays social and economic change, Aoyama Gakuin University makes effort at AML（Aoyama Media Laboratory）project. The author experienced the development of the infrastructure system of that project and adopted the LUCINA method as the development methodology. The project has following features: 1）First, to realize flexibility in heterogeneous environment, the project adopt XML and CORBA as basic technology of data format and communication; 2）Second, a scenario of lesson can't be predefined and repeat itself because collaborating lessons proceed ad hoc. The former causes technological and skills risks, and the latter causes requirements risks.

LUCINA is one of architecture-centric software development processes, which can satisfy changeable requirements and can utilize state-of-art technology. These processes contribute the project management technique, especially risk management, which aims to meet delivery deadline and quality requirements.

This paper sketch out the way of mitigating requirements, skills and technological risks. The author's experiment is good example to show how LUCINA process works.

1. はじめに

青山学院が推進するAML（Aoyama Media Lab.）プロジェクトは，社会・経済構造の変化に対応した新しい教育の方法と基盤の提供を目的としている．日本ユニシスはこの基盤システムを担当し，開発技法LUCINA（旧称：TACT）に基づいて開発した．この報告は，リスク管理に焦点を当ててプロジェクト管理の過程を示す．このプロジェクトの主なリスクは，新しい教育であるため実際の授業を行なうまで要件が固まらないこと，XML（eXtensible Markup Language）などの新規な技術を利用す

るため，技術要素の組み合わせや要員教育の前例が少ないことであった．そして，授業は開始を遅らせられず，やり直しもできない．そのためリスクを確実に解消する必要があった．

2. 当プロジェクトの目的

この章では，基盤システムの課題を明示するため，AML プロジェクトの概要を示す．まず，このプロジェクトが開始された背景と経緯に触れ，次に AML プロジェクトが提供する新教育方法の一端を演習の内容で例示する．

2.1 AML プロジェクトの背景

現在の社会構造の変化に対応した新しい教育（内容，手法，提供方法）の実現の必要性が AML プロジェクトの背景である．ここで社会構造の変化とは，主にグローバル・エコノミーという観点でみた変化であり，市場のネットワーク化と世界規模での競争と連携である．

新しい教育へのニーズの一つは，獲得した知識を活用し，実践的なスキルを身につけるということであり，そのような職業的なスキルを持った学生を社会に送り出すことである．例えば，従来は経営学部での教育は講義およびゼミが中心で，学生にとって必要な知識は習得できても経験を得ることは難しかった．コンピュータを使って実社会を擬似体験することは，それを実現する一つの手段である．このように，情報技術を利用して，新教育へのニーズに応えるシステムを実現するため青山学院大学ではプロジェクトを発足し，活動してきている．

2.2 プロジェクトの経緯

青山学院大学経営学部では，1997 年度から 4 年間の計画で ACC（AOYAMA Cyber Campus）プロジェクトに取り組んでいる．AML プロジェクトは，ACC プロジェクトの成果をベースに，青山学院大学総合研究所における研究活動として推進している．このプロジェクトは，教育方法と教育基盤を開発する．教育方法の開発は新しい教育方法の提案と，そのために必要なデジタル教材や教育用ソフトウェアの開発を行ない，教育基盤の開発はそれらの教育の運用を支援するための教育基盤システムを構築する．

AML プロジェクトの 1999 年度の活動は青山学院大学経営学部，理工学部，文学部および初等部の教員とアシスタント 50 名，システム・ベンダ 4 社の産学共同で実施し，次に示すワーク・パッケージ（以下，WP： Work Package）に分かれて開発し，本番の授業・演習を通じて検証した（表 1）．

教育方法の開発は，経営学部・理工学部・初等部での教育を題材としている．経営学部では，製造業をモデルとしたバーチャルな企業環境を利用して演習を行なう（WP 1〜WP 4）．理工学部では，分散立地したキャンパスで一体化した学習を実現するため，集合教育型の遠隔授業を行なう（WP 5）．文学部では，マルチメディアを利用した「総合的な学習」を題材に，一貫教育を支援するシステムを開発する（WP 6）．一貫教育の特徴は，長期にわたる学習履歴をフォローし，児童・生徒の個別の要求に対応した学習の方向づけすることである．日本ユニシスは，新教育基盤の開発（WP 7）を担当した．

表1　ワーキング・パッケージの構成

WP	教育方法/基盤
WP 1	新製品開発プロジェクト協調演習
WP 2	グローバル CE 協調型演習
WP 3	製販物統合化情報システム協調型演習
WP 4	グローバル SCM 協調型演習
WP 5	集中教育型遠隔授業システムによる教育方法
WP 6	一貫教育におけるマルチメディア型総合教育
WP 7	新教育基盤システム開発

AML プロジェクト全体のマネジメントも日本ユニシスが担当したが，この報告では新教育基盤開発のプロジェクト管理に絞って述べる．

2.3 演習の概要

ここでは，経営学部での演習内容を紹介する．この演習内容は，2.1 節で述べた市場の変化の部分的な鳥瞰図となっている．演習は PC の新製品開発を題材とし，まず新製品を企画し，その製品を仮想企業で開発する．

AML プロジェクトが参考とするスタンフォード大学 PBL（Project Based Learning：協調遠隔型教育方法論とその評価方法）は，建築学科を対象とするため A/E/C（Architecture/Engineering/Construction）というビジネス・プロセスであるが，AML プロジェクトは製造業を対象とするので D/M/M（Design/Manufacturing/Marketing）というモデルを採用している．

新製品企画は，まず市場と商品構造の理解・分析，意匠設計を行なう．この企画をもとに事業計画を立て，マーケティング・リサーチを計画・実施する．一方，この企画を実現するために，エンジニアリング（製品の基本設計）を行なう．エンジニアリングの目標は，開発期間の短縮と品質の向上である．プロセスの並列性を向上させることでこの目標に対応する手法であるコンカレント・エンジニアリングを適用する．このエンジニアリング・プロセスは，設計部門が製品企画部門をはじめ，購買部門，生産部門と連携して実施する．この事業計画とデザインをレビューし，製品の製造・販売に移る．

製品の開発は，企画された商品の部品構成表を基に，キー・コンポーネントごとに部品特性を分析し，部品構成表を改良する．そして，仮想の市場・部品メーカ，配送ルートを利用した，生産・販売・物流を統合したビジネス・プロセスを定義する．さらに，それぞれの業務を，在庫やキャッシュ・フローなどの観点でプロセス全体として最適化するため，SCM（Supply Chain Management）を実施する．

このような演習を行なうために，仮想プロジェクト・チームを編成し，個々の学生が異なった業務を担当する役割を果たすように構成する．全てのビジネス情報はデジタル化されており，統合業務システム（SAP R/3）やプロジェクト・マネジメント・ツール（Siebel 98）等を用いて，仮想的な市場や企業を模擬する．また，グローバルなコンカレント・エンジニアリングを模擬するため，スタンフォード大学と共同演習する．

3. 新教育基盤の要件

これまでの情報技術を用いた教育の試みと比較した場合，AML プロジェクトの特長は，教育方法論の提示と，個々の教材を連携させる基盤の提供である．社会・経済の変化に対応できるように，学生の総合的なスキルを向上させるという教育方法に基づき，さまざまなプロセスが連携した社会を模擬するバーチャルな企業で，協調しながら演習する学習環境を提供している．以下では，2 章で述べたこのような課題に対応した新教育基盤の要件を示す．

3.1 新しい教育基盤の業務要件

新教育基盤システムは，教育方法のさまざまなニーズに応える必要がある．

まず，学習環境の地理的・時間的な制約を無くすことである．学内での複数キャンパス間の連携はもとより，学生の自宅からの利用，国内・国外の他大学との交流も可能にする．そして，講義外の学習や演習，講師への随時の質問などを実現することが要件である．

次に，学習状況のフォローと必要な手助けができることである．そのために，長期にわたって学習履歴を蓄積・保存する．学習履歴は，講義への出席状況，提出物とその評価，成績等である．

3.2 基盤の持つ機能

このような業務要件を満たすため，基盤機能は演習などの授業をサポートする機能と教員の管理業務を支援する機能を提供する（図 1）．

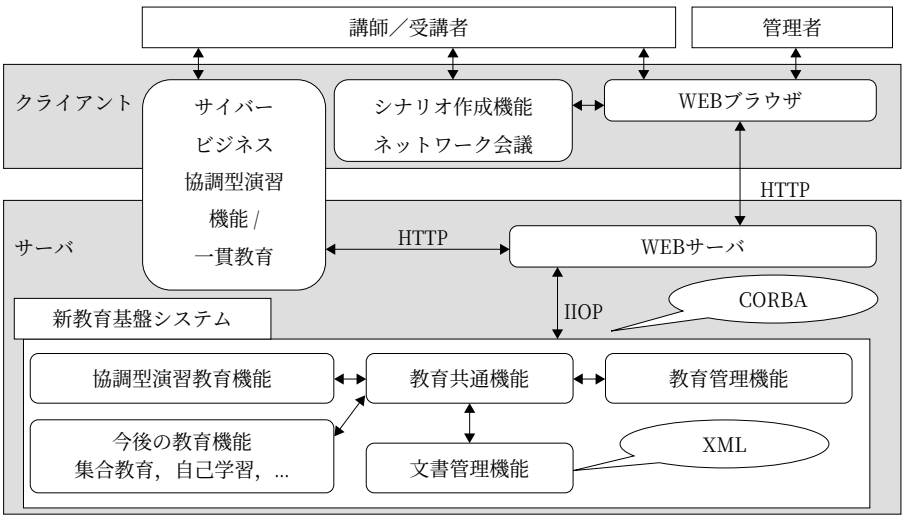

図 1　新教育基盤機能全体図

第 1 の機能は，教育ドキュメントを管理する文書管理機能と講義シナリオを作成するシナリオ・エディタである．文書管理機能は，文書のセキュリティ管理，ファイル・システムとの連携により，デジタル化した教育ドキュメントの効率的な保存，参照，再利用を実現する．シナリオ・エディタは，講義シナリオが持つ意味的な構造（講義名，予習・復習項目，学習項目，参照する教材，提出物など）を利用して，講義シナ

リオを作成するためのツールである．必要に応じた表示抑制機能も提供している．例えば，解答はテスト終了後に表示するようにできる．

第 2 の機能は，各機能の連携であり，基盤の各機能を分散オブジェクトとして提供している．このことで，教材に特化した機能への接続と，各機能の別サーバへの分散配置を容易にしている．

この他，協調型の演習は，協調型演習教育機能と教育共通機能のコラボレーション支援機能（質問箱，電子掲示板，伝言板）及びネットワーク会議機能で支援される．講義に関する情報（講師・受講者，講義日程など），受講者に関する情報（出欠，成績，学習履歴など）は，教育共通機能と教育管理機能で管理される．

以上の機能で，簡易な操作が可能になり，インターネット技術と融合させて地理的・時間的制約を受けずに，多様な形態での情報提供・伝達を実現している．

3.3 先端技術の採用

基盤の機能を実現するためのシステム要件として，CORBA（Common Object Request Broker Architecture），XML，Web などの標準技術の採用がある．このことにより，柔軟なシステム構成と他システム間連携の容易性を狙っている．CORBA，Web，XML と親和性が強い Java を開発言語とした．

まず，デジタル化した教材ドキュメントを長期的に保存し，学外・海外とも交換することを考慮し，文書形式として XML を採用している．XML は文書の共有性向上や標準化を図るための国際標準規格であり，文書構造の定義(DTD：Document Type Definition）と表示方法の指定（スタイル指定）を文書から独立させている．このことにより現在 Web で標準として利用されている HTML（Hyper Text Markup Language）に比べ，文書に論理的な意味を持たせることができ，検索性を高めることができる．この XML の利点を生かし，長期間にわたる保存・改訂の容易性，セキュリティ管理などを実現するために XDS（XML Document Server）を開発した．

そしてデジタル教材を有機的に結び付けるために，CORBA を採用している．CORBA は分散オブジェクトの国際標準であり，特にオブジェクト間の通信プロトコル（IIOP：Internet Inter-Orb Protocol）は，複数のシステム間を連携するためのデファクト・スタンダードになっている．デジタル教材は，マルチメディア・データはもとより，SAP R/3 などのアプリケーションとして提供される．一貫教育や集合学習・自己学習の教材も同様である．

3.4 短期間での開発

今回の開発は，単年度での開発・実証が必要であった．後期（9 月〜1 月）の講義で利用する，4 月に要件が固まる，という条件から実質的な開発期間は，6〜8 月の 3 ヶ月間である．4，5 月で要件をシステム的に定義・分析し，9 月は他のアプリケーションとの統合テストや実際の授業シナリオによるシステム・テストの期間と予定された．

規模や複雑さの課題は少ない開発であるが，先端技術を採用するため対応できる要員を揃えることと，講義のやり直しはできないため納期と品質を確実に守ることが要求された．

4. LUCINA の概要

3 章で述べた要件を持つプロジェクトに対応する開発技法として LUCINA を採用した．LUCINA は日本ユニシスが提供するコンポーネント・ベースの開発技法であり，アーキテクチャ中心の開発プロセスを特徴としている．

アーキテクチャ中心の開発プロセスは，要求はいつまでも固まらない，紙上の検討では正しい設計が得られないこと[6]を前提として，プロジェクトを成功させることを目指している．今回のプロジェクトは，新しい教育方法を対象としている，XML など利用技術が未成熟な IT を利用する，短期間でやり直しが利かない，などの特性により，このような開発プロセスを必要とした．

この章では，プロジェクト管理の観点からみた LUCINA の開発プロセスを紹介する．そのために，まず同種の開発プロセスの代表である RUP（Rational Unified Process）の概略を示し，それとの比較で LUCINA の特徴を述べる．

データ保全性の重視，ユースケース[*1] 主導開発など，LUCINA のもつ他の特徴は，著者の別の報告[5]で紹介している．

4.1 プロジェクト管理とリスク駆動開発

適切なタイミングでサービス・製品を投入することを競争力の源とするビジネスに対応し，ビジネス・アプリケーション開発は，決められた日にサービスを開始できるよう，短期間で確実に稼働させることが求められている．

これを実現するためには，開発期間の期待値とその偏差を最小化する必要がある．プロジェクト管理は計画の管理であり，ここでは計画した開発期間を期待値，計画通りに終わらない確率を偏差と見る．期待値を小さくする，すなわち短期間に完成するためには，生産する規模を小さくすることと，生産性を上げるという方法がある．このための基本的な手段は，再利用率を上げることと，開発組織の効率を向上させることである．一方，開発期間の偏りを少なくするために必要なことは，開発上のリスクを制御できるようにすることである．

Booch[1]は，開発プロセスに関して取り組む必要がある核心の問題として，次のポイントを示している．“ プロジェクトにおいて，一人一人のプログラマにおける創造性へのニーズと，管理部門における安定感と先見性に対するニーズを，どのようにして両立させればよいのであろうか ”．これは，リスクの管理と大きく関係している．優秀なプログラマの創造性は，さまざまな課題を解決し，いくつものリスクを無くしていく．一方，創造性に頼る開発は，スケジュールが予見しがたいというリスクを持ち，不安定となる．

以下では偏差を小さくすること，すなわちプロジェクト管理のリスク管理に焦点を当てる．リスクへの対応策（Risk Response）は回避（avoidance），軽減（mitigation），受容（acceptance）の 3 種類がある[4]．ライフサイクルの各フェーズの目的を，当面する最重要なリスクに着手し，軽減させることに置く開発プロセスを，リスク駆動（risk-driven）開発と呼ぶ．

4.2 アーキテクチャ中心の開発プロセス

リスク駆動開発を実現するため，いくつかの開発プロセスが提案されている．Rational 社は，開発プロセスのフレームワークとして RUP[6]を定めており，影響力のあ

るプロセスの一つである．RUP の標語は，アーキテクチャ中心，追加的(incremental)で反復的（iterative）な開発，ユースケース主導開発である．ここでは，RUP と共通する特徴をもつ開発プロセスを，アーキテクチャ中心の開発プロセスと呼ぶ．この節では，アーキテクチャ中心という用語の意味を整理する．

4.2.1 アーキテクチャ

これまでいくつかアーキテクチャの技術的な定義がなされてきた. 例えば, Booch[1] はアーキテクチャを構成するパーツとして基本オペレーティングシステム，ネットワーキング，永続オブジェクトの記憶，分散オブジェクト管理，アプリケーション環境，ドメイン非依存のフレームワーク，ドメイン特有のフレームワークおよびドメイン・モデルを挙げている．しかし，一般に認められた定義はない．RUP ではアーキテクチャの技術的な定義はシステムの構造・振る舞い・パターンとするにとどめている．プロジェクト管理の観点からすれば，開発するシステムのうち何をアーキテクチャに含めるかはシステムの特性とリスクの見極めであり，これこそがプロジェクト管理の要点である[8]としている．

プロジェクト管理の観点から見た場合の用語は, アーキテクチャ, アーキテクチャ・ベースライン（Architectural baseline），アーキテクチャ記述（Architecture description）である．アーキテクチャはプロジェクトのビジョンであり，プロジェクトで開発するシステムの主要な特性を含む．ただし，アーキテクチャは開発成果物としての実体はない．アーキテクチャ・ベースラインは, 構築フェーズのベースラインとなる．ベースラインは構成管理の用語であり，レビューされた実行可能なリリースで，要件から設計・実装・導入までの一連の成果物を含む．そして，アーキテクチャで表現されたビジョンがどのようにアーキテクチャ・ベースラインとして具体化したかを記述するのが, アーキテクチャ記述である. アーキテクチャ記述は設計成果物に含まれる．

4.2.2 アーキテクチャ中心

アーキテクチャ中心とは，アーキテクチャを記述し，構成するための道具としてモデルを捉え，開発プロセスを実行可能なアーキテクチャを次々と洗練することと捉えることである．

アーキテクチャ中心のプロセスは，いくつかのフェーズに従って進行するが，このフェーズをシステムの状態によって区別している．システムの状態とは，プロダクトのリリースの段階であり，リリースとはシステムの安定した，自己完結性のある，実行可能なバージョンのことである[1]．一回のリリースを生み出す開発作業を反復（iteration）と呼ぶ．

アーキテクチャ中心のプロセスが必要となる第一の理由は，開発が反復型とならざるを得ないことである．RUP では，ユーザの要求はいずれ固まると仮定すること，紙上の検討で実装以前に正しい設計が選られると仮定することは，いずれも誤り[6]とし，そのため繰り返し開発が妥当である，としている．ウォータフォール・モデルはこの仮定に立脚しているため，誤りはすべて実装時に起こり，開発の終盤に一斉に行なう統合テストで発見された誤りは容易に修正できるという前提を持つ[13]．しかし，現実にはユーザの要求は不安定であり，ユーザが利用を始めるまで変更はありえる．また，ビッグ・バン統合といわれる開発最終盤での結合・統合テストのリスクが高い

ことは事実である．また，機能的な要件は追跡可能であり検証可能だが，操作性やリソースの競合，性能のボトルネック，制御の競合などを含めた設計の正しさは紙上で検証できない[8]，と考える．従って開発は反復するという主張は妥当である．そして，最終的なリリースへ確実に進捗するためには，開発のリスクを制御する必要がある．反復が作り直しの連続にならないためには，開発プロセスをアーキテクチャの洗練としてすすめる方法が必要となる，と考える．

4.2.3 マクロ・プロセスとマイクロ・プロセス

4.2.5 項で触れるが，リリースの反復によるプロセスは，スパイラルな開発の発展形である．RUP は，開発プロセスをマクロ・プロセスとマイクロ・プロセスに分け，マクロ・プロセスで各々のリリースをマイルストーンとして進捗を管理し，マイクロ・プロセスでウォータフォール・モデルと同様の工程で管理することを求めている（図 2）．

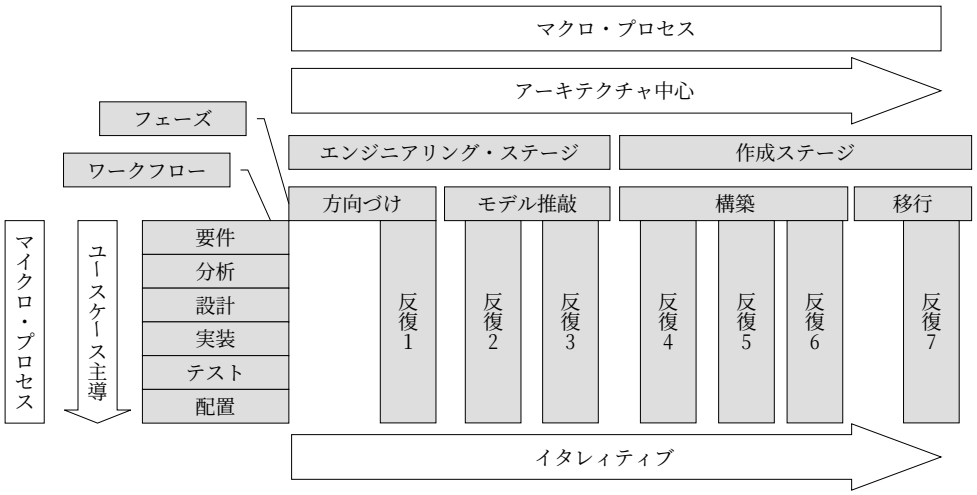

図 2　RUP のマクロ・プロセスとマイクロ・プロセス

マクロ・プロセスはプロダクトのリリースの進行に合わせたフェーズと反復で構成される．まずシステムの特性をアーキテクチャとして示し，構築作業の起点となるアーキテクチャ・ベースラインを作成する．構築した結果を受け入れテストして修正し，最終的なリリースを行なう．このようにフェーズは，方向づけ（inception）・モデル推敲（elaboration）・構築（construction）・移行（transition）に分かれ，この流れを開発ライフサイクルと呼ぶ．各フェーズの中で，複数の反復（iteration）が行われる．各々のフェーズの最後に置く主要マイルストーン（major　milestone）はレビュー・ポイントであり，リリースの条件を示している（表 2）．

一方，マイクロ・プロセスは，ワークフロー（process workflow）で構成される．それぞれのワークフローは，複数の作業（Activity）に分割される．核となるワークフロー（core process workflow）は，要件定義（requirements），分析・設計（analysis/design），実装（implementation），テスト（test），配置（deployment）であり，

表2　アーキテクチャ中心とリスク駆動の関係

	方向づけ	推敲	作成	移行
リスクの推移 リリース	リスクの探索 プロトタイプ	リスクの解消 アーキテクチャ・ベースライン	低リスク 使用可能なリリース	低リスク 製品リリース
Boehm のマイルストーン COCOMO II の類型	Lifecycle Objectives (LCO) アプリケーション組み立てモデル	Lifecycle Architecture (LCA) 初期設計モデル	Initial Operational Capability (IOC) ポスト・アーキテクチャモデル	

ウォータフォール・モデルの開発工程に対応する．個々の反復は，マイクロ・プロセスに従って，ユースケース主導で行われる．

この意味では，ウォータフォール・モデルはリリースが一回のプロセスといえ，マイクロ・プロセスが開発ライフサイクルとなる．このライフサイクルでは文書に基づくレビューによって，開発プロセスが進行する．例えば，要件を 100% 定義できたところで，分析工程に進む．これに対し，アーキテクチャ中心の開発プロセスでは，文書による記述を必要最小限にし，実証可能なリリースによる管理を目指している．

なお，ライフサイクルの前半（方向づけ・モデル推敲）をエンジニアリング・ステージ，後半（構築・移行）を製造ステージと呼んでいる．

4.2.4　リスク駆動開発との関係

表 2 に示すように，このアーキテクチャ中心のプロセスは，リスク主導の開発と対応する．RUP の主要マイルストーンは，Boehm[9]の重要マイルストーン(critical milestone）を採用している．ウォータフォール・モデル，スパイラル開発モデルの双方への反省に基づき，成功したスパイラル開発の事例からこのマイルストーンは提案され，米国防総省（DoD：US Dept. of Defense）の Stars プロジェクトなどで実証されてきた．

Bohem は，COCOMO（Constructive Cost Model）の提唱者でもあり，COCOMO II はこの重要マイルストーンと対応している．COCOMO は開発規模とコストの関係を推定するモデルであり，COCOMO II はそれを補強したものである．COCOMO II のスケール誘因（scale driver）は，開発規模に対しコストを指数的に増やす原因であり，プロジェクトのリスクとなる．アーキテクチャ中心の開発プロセスは，この誘因への対応策ともなっている[7]．特に，開発の先例性とリスクの解決という誘因に関しては，2 回目の反復では馴染みのある開発となり，エンジニアリング・ステージでアーキテクチャや重要なリスクを完全に解決することを目標としている．

4.3　LUCINA のフェーズとリスクへの対応

ここでは，プロジェクト管理の観点から，RUP との対比で LUCINA の特徴を紹介する．スパイラル開発は大規模な開発へ適用する場合不安定である[11]という指摘がある．LUCINA はアーキテクチャの事前定義によってアーキテクチャ中心をより推し進め，リスクの解消を容易にし，プロジェクトを管理可能なものにし，技法に具体性を持たせている．

4.3.1　LUCINA のプロセス

他のアーキテクチャ中心の開発プロセスと同様，LUCINA はシステムを本格的に構築するフェーズに入るまでに重要なリスクを解消し，システムを構築するフェーズ

のリスクを最小化する，というアプローチを取る．LUCINA の開発プロセスは RUP と同様に，立ち上げ・練り上げ・作り込み・持ち込みの四つのフェーズからなる（図3）．このフェーズに従って，プロジェクトを立ち上げ，アーキテクチャと要件を練り上げ，システムを作り込み，実行・運用環境に持ち込む．

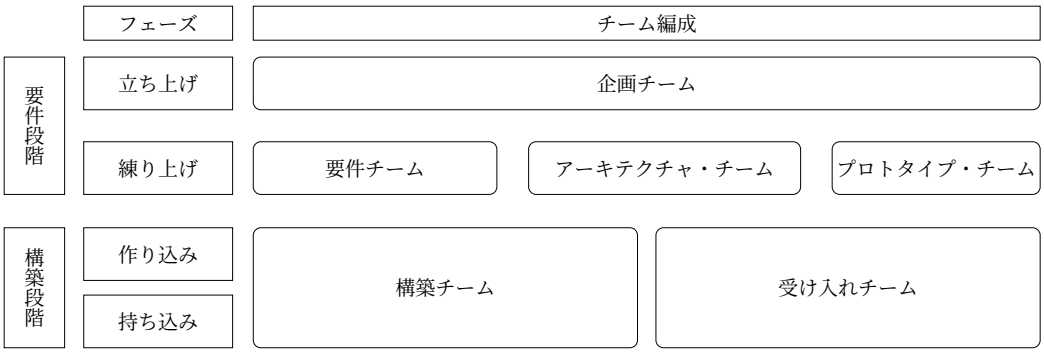

図 3　LUCINA のフェーズとチーム編成

練り上げフェーズは，要件チーム，アーキテクチャ・チーム，プロトタイプ・チームに分かれて行なう．要件チームは，機能的な要件と非機能的な要件を固める．機能的な要件は，振る舞いをユースケース・モデルなどを利用して記述する．アーキテクチャ・チームはアーキテクチャ記述を作成する．プロトタイプ・チームはこの要件定義，アーキテクチャ設計と並行したプロトタイプ作成により，要件を実現できること，パフォーマンスなど非機能的な要件を満足できることを検証し，アーキテクチャ・ベースラインを作成する．このようにして，LUCINA の開発プロセスでは，練り上げフェーズでリスクを解消し，作り込みフェーズ以降ではリスクの少ない開発を実現する．創造性は練り上げフェーズで発揮し，作り込みフェーズ以降では，予測可能なスケジュールで限られた創造性しか必要としない作業を行なう．

4.3.2　アーキテクチャ中心からアーキテクチャの事前定義へ

グリーン・フィールド開発と呼ばれる一回目の反復は難しい[14]といわれる．実際には，共通的な基盤を設計する作業は技術的なリスクとなり，練り上げフェーズ内で実行可能なアーキテクチャを築くことはスケジュール上の制約に合わない可能性がある．LUCINA ではアーキテクチャを事前提供することにより，練り上げのフェーズ初期から要件に基づくプロトタイプを実行でき，アーキテクチャの実現可能性（feasibility）と要件充足性（sufficiency）を確認することができるよう，工夫している．

LUCINA の特徴であるアーキテクチャの事前定義とは，アーキテクチャ・パターンを基にした実行環境・開発環境のプロファイルを示し，そのプロファイルに対応した設計方針や手順，ツールの利用技術を具体的に定義することであり，その定義を前提として開発プロセスをすすめる．この点でも，アーキテクチャの構想から始める建築型ではなく，共通のプラットフォーム上に部品を組み立てる製造業型のプロセスといえる．

今回採用した CORBA と Java を使用する分散オブジェクト・パターンのアーキテ

クチャ[12]は，Booch の分類[1]に従えば表 3 の通りである．

表 3　アーキテクチャの構成要素

構成パーツ	プロダクト
基本オペレーティングシステム	Windows NT
ネットワーキング	HTTP/IIOP
永続オブジェクトの記憶	SYSTEM ν［nju：］，Oracle
分散オブジェクト管理	SYSTEM ν［nju：］
アプリケーション環境	Java 2，Internet Explorer
ドメイン非依存のフレームワーク	SEWB＋，Web Page Generator
ドメイン特有のフレームワーク	XDS

プロファイルに基づく作業の手順は，具体的に技術を習得でき，学習曲線の立ち上がりを早くすることを目指している．作業の手順は，オブジェクトの導出やプログラミングなど，要件から実装まで一貫した事例やツールを利用した導出手順を明示している．

4.3.3　マクロ・プロセスの管理

マイクロ・プロセスに含まれる個々のワークフローから見て，マクロ・プロセスのフェーズ間の境界は曖昧なため，作業が直列的に進むウォータフォール・モデルと比べ管理は難しくなる．RUP はツールによるプロセスの自動化で，この困難を回避しようとしている．

これまで反復という言葉で一括してきたが，繰り返しには反復的（iterative）と追加的（incremental）の二つの側面がある．反復的な開発は反復を繰り返し，個々の反復はリリースとリリースの間の期間を表す．追加的な開発はベースラインに実行モジュールを追加してシステムを組み上げていく．追加の作業が並列的に行われる場合，追加作業は複数の反復にまたがることもある．そのため，反復と追加の関係はさらに工夫を要する複雑なものとなる[8]．

LUCINA の特徴であるコンポーネント・ベースの開発技法は，コンポーネントを調達し，組み立ててアプリケーションを作成する技法である．コンポーネントが充分揃っていない現状では，調達の方法は再利用のみならず，カスタム開発を含む．コンポーネントの基本的な定義は独立して開発できる単位であるので，コンポーネントは反復よりも追加（increment）の単位となる．

RUP が示すマクロ・プロセスとマイクロ・プロセスの関係は，ワークフローを縦軸，フェーズを横軸とした作業量の分布として図示される．この分布のカーブをここでは「波」と呼ぶ．

この複雑さを解消するために，LUCINA は個々の追加作業に対応させて「波」を分解している．まず練り上げフェーズでは，各々のプロトタイプ作成が反復であり，追加である．このフェーズではアーキテクチャの事前定義と反復の数の指定によりアーキテクチャ・ベースラインを構築する作業を明確化し，それを担うアーキテクチャ・チームとプロトタイプ・チームの作業を別ワークフローとして独立させている．作り込みフェーズでは，独立に実行できる単位であるコンポーネント・システム[10]を

追加の単位に対応させ，キーとなるコンポーネント・システムのリリースに合わせて反復を決めている．この結果，それぞれの「波」がほとんど特定のフェーズに入っている．このため，既存のプロジェクト管理体系のレビュー・ポイントと対応づけが容易になり，発注単位も明確になっている．

5. AML プロジェクトのリスクとその管理

先に述べたように，アーキテクチャ中心のプロセスは，新しい技術の適用，要件の不安定さに対応でき，この点で LUCINA のプロセスは AML プロジェクトのシステム開発に適した構造を持っている．

ソフトウェア開発上のリスクは SEI による分類[2]もあるが，Fowler ら[14]は，要求，技術，スキル，政治に分類している．要求リスクは要件を把握でき，それを実現できるか，技術リスクは利用する IT がうまく組合わせられるか，スキルリスクは必要な人員や専門家が集められるか，政治リスクはプロジェクト外から割り込んでくる圧力があるか，である．

3 章で述べたように，このプロジェクトは次の二つの特性を持つ．まず，協調的な演習を中心とすることから，授業はやり直すことができず，利用シナリオを完全には事前定義できないこと．そして，多様な環境に対応する柔軟性を持たせるため，基盤システムのデータと通信の技術として XML と CORBA を採用したこと．前者の特性は要求に関するリスクを，後者はスキル・リスクや技術的なリスクをもたらす．

この章では，要求リスク，スキル・リスク，技術リスクに対応した過程を，不確実な要件に対応する，先例性の無い技術に取り組む，アーキテクチャ先行でリスクを解消していく，という点から具体的に示す．

5.1 要求リスク

AML プロジェクトのように新しい技術を使う業務は，新しい業務形態を導入する場合が多い．この場合，新しい業務形態は，まだ現実のシナリオを経験していないため，要件が従来業務以上に不確実となる．このようなプロジェクトに対応するためには，要件を正しく認識することと，頑健なアーキテクチャの作成が課題となる．ここでは変更に対応する能力の高いアーキテクチャのことを，頑健なアーキテクチャと呼ぶ．

LUCINA では要件を認識するために，シナリオおよび高レベル・シーケンス図を利用する．

5.1.1 シナリオの意味

まず，要件を記述するために用いるシナリオと高レベル・シーケンス図の意味を紹介する．LUCINA では，開発作業は，対象が問題空間（problem space）か，解空間か（solution space）か，システムの外部モデルか内部モデルか，という軸で分割されている（表 4），と捉える．問題空間はシステムの要件と制約であり，解空間は実行可能な製品を表現する．要件定義は要件をシステム外部のモデルとして記述し，論理設計はその要件を内部のモデルに割当てる．LUCINA はオブジェクト指向技術の適用範囲をアーキテクチャと内部モデルに限定している．外部モデルはシステムの振る舞いを操作的に記述する．操作的な記述はオブジェクト指向と親和性が高いが，内

部モデルがオブジェクト指向であるという制約は持たない．LUCINA では，このシステムの振る舞いをシーケンス図で描く．このシーケンス図を高レベル・シーケンス図とよび，シナリオを図として表示したものである．

表 4　各作業の位置づけ

モデル / 空間	外部	内部
問題	要件定義	分析
解	アーキテクチャ構築	
	—	設計

高レベル・シーケンス図の第一の意義は，個々のユースケースを構成するシナリオの形式的な記述である．必要性を指摘し，表現法を提案している開発方法は多いが，シナリオの形式的な記述方式は未だ統一されていない．

高レベル・シーケンス図の第二の意義は，複数のユースケースに跨るビジネス・プロセスの記述である．ユースケースの構造化（《include》《extend》）は，特定のユースケースに含まれるシナリオの構造化であり，複数のユースケースを複合したビジネス・プロセスを表現できない．また，ユースケースの形式的な定義を変更して，ユースケースからユースケースを呼び出す記法を取り入れている方法もあるが，これも具体的なシナリオを示すことはできない．このような仕組みにより，ユースケースとそのシナリオでユーザの要求を確認することを目指している．シナリオは，統合テスト・システムテストでも利用する．

5.1.2　シナリオの有効性

このプロジェクトでは，新しい教育を実現することを目的としている．新しい，という意味は教育手段をできるだけ電子化するということと，シミュレーションを主体とした演習を行なうということである．

集合学習に見られるような，On Demand の教育形態は，CAI（Computer Aided Instruction）以来の実績があり，従来のマスプロ教育の延長である．しかし，シミュレーションを中心とした演習は，少なくとも日本の経営学教育においては，前例がほとんど無い．

練り上げフェーズで，基盤開発チームでは高レベル・シーケンス図を用いて教育シナリオを描き，そのシナリオと照らして機能の過不足をチェックしていた．

基本機能をリリースして持ち込みフェーズに入ってからも，教育方法開発チームは授業に利用する教育シナリオを記述しきれなかった．このシナリオは，複数の教員による複数の講義のテンプレートとなり，その中で学習が効果的になされ，効果の測定に必要なデータを獲得できるようにしてある必要がある．

教育シナリオを作成する上での障害を次のように推測した．まず，教育シナリオを作るということは，日常の教育活動には無い作業である．大学の教員は，講義のストーリである教育シラバスをそれぞれに作成するが，講義の場での個々のやり取りは明示化されず，また共通化されたものでもない．また，教育シナリオ作りは本質的に創

造的な仕事である．通常のソフトウェアと同様で，一度業務として使ってみれば操作方法を理解できるが，たとえ操作できるものであっても機能だけ与えられたのでは，それをどのように組み合わせて業務として組み立てるかを案出するのは難しい．

そこで，練り上げフェーズで作成した高レベル・シーケンス図を，教育方法開発チームに提示して，教育シナリオを作成した．図 4 は高レベル・シーケンス図の例である．このシナリオは，講師の指導で学生が手引書を入手し，それに従って演習内容を決め，パラメータを決定してシミュレーションを行ない，その結果を検討する，という演習のプロセスを，アクター（講師，受講者）とユースケース（文書管理，質問，掲示）のやり取りとして記述している．このように，高レベル・シーケンス図は，直観的に理解できるものであり，図の表記法を説明する必要はなく，すぐに理解された．また，この図は基盤開発チームが主体となって，内容を協議し合意するのに有効であった．

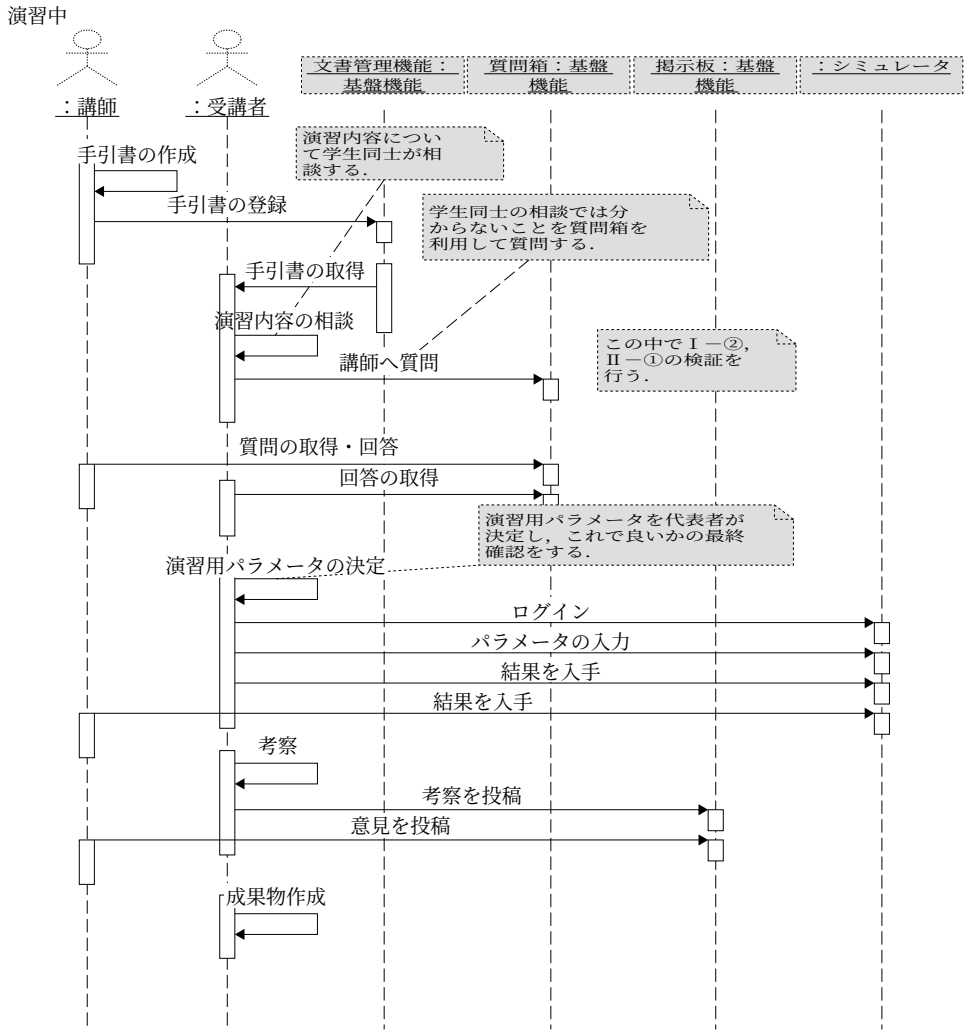

図 4　高レベル・シーケンス図

このようにして客先の要求を再度明確化し授業を行なった．持ち込みフェーズ以後での機能変更・追加をアーキテクチャが吸収できたことから練り上げフェーズで高レベル・シーケンス図を利用したことは，以下の 2 点で頑健なアーキテクチャの提供に貢献した，と考えている．まず複数のユースケースをまたがるシナリオを描くことで，個々の機能に限定されないアーキテクチャを検討でき，操作をユーザの視点で考慮できたという点である．また，持ち込みフェーズでの高レベル・シーケンス図の利用は，要件を明確化することうえでこの図が有効であることを示している．

残された課題は，練り上げフェーズで高レベル・シーケンス図を提示して，早期に要件を固める方法と，要求側でこの図を描けるよう指導する方法を確立することである．

5.2 スキル・リスク

開発の先例性（precedentedness）は，COCOMO II モデルでは，スケール誘因の一つとして取り上げられている．開発の先例性は，関連するシステムの開発経験があるか，革新的な技術を必要とするかしないか，などの先例性の度合いを示す．

5.2.1 先例性の意味

LUCINA では次のようにして，構築フェーズへ入ったところで，その開発が先例性のあるものとなるように工夫している．

4.3.1 項で示した開発プロセスに従い，練り上げフェーズで実行可能なアーキテクチャを実装する．このアーキテクチャ上で作成されたプロトタイプ・システムは，構築フェーズでのサンプル・コーディングとなる．

そこで先例性は，LUCINA に含まれない要素技術に対するものと，LUCINA に含まれるが開発者にとって馴染みの無いものの 2 種類に分かれる．ここでは LUCINA に含まれるが開発者にとって先例性の無い技術に対する対処方法を述べる．

開発が始まる前，アーキテクチャ・チームのメンバーはこれらの要素に経験を持っていたが，構築チームでは，Java の経験者は約 2 割，CORBA，XML に関しては全く経験がなかった．従って，今回の開発チームにとって先例性のない，あるいは少ない技術は，CORBA，Java，XML およびビデオ会議である．このうち LUCINA に含まれる要素は CORBA と Java であり，含まれない要素は XML とビデオ会議である．

5.2.2 先例性への対処

LUCINA は事前定義されたアーキテクチャとプロファイルに対応する技術，教育手段を整備している．今回のアーキテクチャである分散オブジェクト・パターンに対応するプロファイルには CORBA，Java が含まれている．ツールの操作を含めた LUCINA のガイドは約 200 ページであり，この資料を渡した時点では効果的な学習ができなかった．それは，以下のような理由によると考えた．まずは量が多いことである．参照マニュアルとしての要素もあるので初学者には情報が多い，3 日程度の教育コースを前提にしているのでひと通り操作するのに時間が掛かる．また，例題は実際開発するアプリケーションとは異なるため，置き換えが必要なことである．

そこで，開発チーム内でスタータ・キットとして，5 枚の手順書が作成された．このキットに従って，一通り操作すれば，一日程度でこのプロジェクトに必要な新規技術に触れることができるというものであった．また，このキットの題材そのものが小

さな独立した一つのモデルとなっている．このキットを利用した後は，ガイドを参照しつつ，順調に開発が進んだ．プログラマにとっての先例性は，対象とする技術を利用して実行可能なプログラムを作成することでほとんど達成された．あとは，アーキテクチャと設計方針（design strategy）に従い，プロトタイプ・チームの助言とマニュアルを利用して技術を高めていった．

これは一つの経験に過ぎないが，この事例からスキル・リスクを軽減させる方策が推測できる．その方策とは，このような量のスタータ・キットを作成させ，練り上げフェーズで実施することと，また新技術を採用する場合，このような量のキットで一通り体験できる範囲内に納めることである．リスク管理の観点で見れば，このような作業をプロジェクトの計画に入れておくことでリスクを軽減できる，と捉えられる．

5.3 技術リスク

5.3.1 XML

XML は LUCINA にとって新規技術であった．XML は，SGML など従来のメタ言語に比べ操作しやすい言語となっている．特に Java を開発言語とした場合，XML Parser が多数普及している．そこで XML を利用したアプリケーションを作成する場合，DTD の定義とその DTD に基づく文書を保存する効率的で拡張性のある永続化を技術的なリスクとした．この部分は，汎用的な XML ドキュメントシステムを XDS として，独立したコンポーネント・システムとして開発した．

この XDS をキーとなるコンポーネント・システムと捉え，XDS のリリース段階に対応して，作り込みフェーズの反復を設定した．但し，XDS の開発プロセスは業務システム開発を対象とした LUCINA の範囲外であり，今回の考察の対象外である．

5.3.2 アーキテクチャ・プロトタイプ

このような対処をした上で，LUCINA に従ったリスクの解決を行なった．

LUCINA では，実行可能なアーキテクチャを確立するためにプロトタイプを作成する．LUCINA の開発プロセスは，検証済みの事前定義されたアーキテクチャを前提とするが，プロジェクト毎にプロトタイプによる検証を行なう．それは，要件が異なることから，ドメイン依存の部分，トレード・オフとなる要素に関するプライオリティ，重要なユースケース，物理的な配置などがプロジェクトごとに異なるからである．

アーキテクチャの検討の結果を経て，アーキテクチャ・プロトタイプ(architectural prototype）を作成した．一度目は実行可能性を検証し，二度目はパフォーマンス要件を満たせるかを検証した．この検証は，最重要なユースケースに基づき行なった．最重要なユースケースとは,アーキテクチャに影響を最も与えるユースケースである．

具体的には，文書の登録・検索・取得が最重要なユースケースである．文書に関する処理は，新規技術を用いることと，大量のデータ転送がありえるためである．XDS の開発と並行して行なったため，プロトタイプは XDS のインタフェース定義に適合するコンポーネントで代用して検証した．

5.3.3 検　　証

このようなリスク軽減策の有効性を全行程の障害累積による信頼度成長曲線（図 5）で検証する．なお，練り上げフェーズはアーキテクチャ・ベースライン構築中なので，

バグの履歴はない．

検証は，①新規技術，②アプリケーション，③開発基盤のカテゴリで行なう（図 5 では，それぞれ XDS，AP，infra と表示）．開発基盤と新規技術がアーキテクチャ・ベースラインを構成する．新規技術に関連する部分をコンポーネント・システムとして切り出したため，そのコンポーネント・システム及び利用アプリケーションの双方の開発を並行できた．新規技術がベースライン化したのは，リリース 2 の時点である．信頼度成長曲線をみると，検証対象としたアーキテクチャ・ベースラインは，7 割以上のバグは作り込みフェーズの前半で出て，その後安定している．

アプリケーションは典型的な信頼度成長曲線を示している．作り込みフェーズでのアーキテクチャ・ベースライン利用技術に関する障害は，プロトタイプ・チームが聞き込みに回ったところから，はっきりと減少した．このような運用は，プロジェクト管理上有効であった．持ち込みフェーズでの障害は，アーキテクチャ・ベースラインに影響を与えない軽微なものであった．また，運用に入ってからのバグは僅かであった．

この結果，リスク対応策は有効であったと考える．練り上げフェーズでの，アーキテクチャ・プロトタイプ作成は構築時の障害の収束を早めた．特に持ち込みフェーズ以降でアーキテクチャ・ベースラインの変更が無かったことは，アーキテクチャと要件定義が有効であったことを裏付けている，と考える．

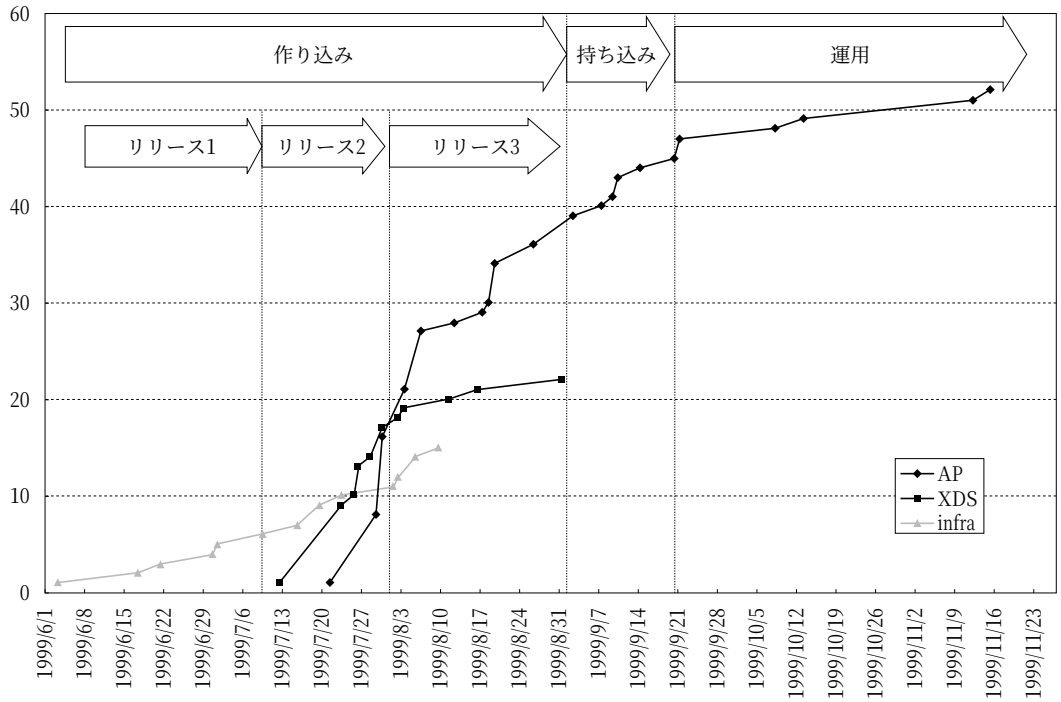

図 5　信頼度成長曲線

6. おわりに

青山学院 AML プロジェクトは，通産省および情報処理振興事業協会における情報化教育モデル学習システム構築事業の一つである．このプロジェクトは 1999 年度下期の授業・演習を通じて検証を終えた．今後の展開としては，AML プロジェクトの実証実験結果を踏まえて，より多くの授業運営に適用し，青山学院大学全体や社会人教育などに拡大し，さらには，他大学への普及を目指している．

今回のシステム構築を通じて，プロジェクト管理に対する LUCINA の有効性も確かめられた．まず，アーキテクチャを事前定義しているため，何が新規技術かを見極めることができた．また，そのアーキテクチャを土台にすすめるため，新規技術を採用する場合に必要となる，実行可能なアーキテクチャによるリスクの検証も低コストで実施できた．そして，コンポーネント指向に基づき，外部モデルを重視していることは，要件の確認に有効であった．

一方，新規な技術に対する方策は，創造的な作業であるため一般化できない部分もある．今回は，XML 部分のコンポーネント化，スタータ・キットの作成による教育などにより対処した．現実に起こるさまざまな課題への解決策は過去の事例を参考にしていくしかない．LUCINA の枠組みで開発を進めることで，このような方策や事例をより有効に活用できる，と考えている．

＊1 ユースケース（use case）：ユースケース・モデルはシステムの使い方（way of use）を示す．ユースケース・モデルは，システムの機能を表わすユースケースおよびそのユースケースを用いるアクターで構成される．アクターは，システムの環境であり，ユーザやプリンター，他システムなどを表現する．業務システムの場合，個々のユースケースは，一つのプリミティブな業務プロセスに対応する．そして，一つのユースケースは，一つ以上の業務シナリオで表現される．

ユースケース主導開発とは，このユースケースとシナリオに対する追跡可能性を保証しながらすすめる開発プロセスのことである．プロジェクト管理の観点でいえば，ユースケースを検証（verify, validation）の道具として要求管理・品質管理を実施することに相当する．

参考文献

［1］ G. Booch,“ Object Solutions : Managing the Object-Oriented Project ”, Addiwon-Wesley, 1996［邦訳：石川克巳訳, オブジェクト・ソリューション, アジソン・ウェスレイ, 1998］

［2］ Sisti, Frank J., and Sujoe Joseph,“ Software Risk Evaluation Method, Version 1.1 ”, SEI Technical Report, December 1994

［3］“ COCOMO II Model Definition Manual ”, University fo Southern California, 1998.（http : //sunset.usc.edu/COCOMOII/Cocomo.html）

［4］ Project Management Institute, A Guide to the Project Management Body of Knowledge, 1996［邦訳：エンジニアリング振興協会訳, プロジェクトマネジメントの基礎知識体系, 1997, 三州社］

［5］ 羽田昭裕, コンポーネント指向の Java アプリケーション開発技法, 技報 63 号, 日本ユニシス, 1999

［6］ P. Kruchten,“ The Rational Unified Process ”, Addison-Wesley, 1998.［藤井拓監訳, ラショナル統一プロセス入門, ピアソン・エデュケーション, 1999］

［7］ OMG Unified Modeling Language Specification, Object Management Group, 1998

［8］ W. E. Royce,“ Software Project Management : A Unified Framework ”, Addison-Wesley, 1998.［邦訳：日本ラショナルソフトウェア監訳, ソフトウェアプロジェクト管理, アジソン・ウェスレイ, 1999］

［9］ B. Boehm, Anchoring the Software Processes, IEEE Software, July, 1996.（http : //sunset.usc.edu/COCOMOII/cocomo.html）

[10] I. Jacobson, et al, Software Reuse, ACM Press, 1997.［邦訳：杉本ら訳, ソフトウェア再利用ガイドブック, トッパン, 1999］
[11] 高橋宗雄, クライアント/サーバシステム開発の工数見積もり技法, ソフト・リサーチ・センター, 1998
[12] 溝上昌宏, Java/CORBA アプリケーション開発環境の実現, 技報 63 号, 日本ユニシス, 1999
[13] F. P. Brooks, Jr., The Mythical Man-Month, Addison-Wesley, 1995.［邦訳：滝沢ら訳, 人月の神話, アディソン・ウェスレイ, 1998］
[14] M. Fowler, K. Scott, UML Distilled, Addison-Wesley, 1997.［邦訳：羽生田監訳, UML モデリングのエッセンス, アジソン・ウェスレイ, 1998］

執筆者紹介 羽 田 昭 裕（Akihiro Hada）

1984 年一橋大学卒業．同年日本ユニシス(株)入社．意思決定支援ソフトウェアの開発・適用に従事．その後，業務システムとその基盤の要求分析・開発に従事．現在，E ビジネス技術部コンポーネント技術室に所属．情報処理学会会員．