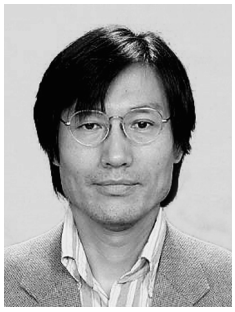
# 情報システムサイクルとパタンランゲージ*1



第３事業部
担当部長

児玉　公信

**Kiminobu　Kodama**
kiminobu-kodama@exa-corp.co.jp

企業情報システムが効果的であるためには、施主である企業の情報システム部門自身が、情報システムがどうあるべきかを明確に提示する必要がある。これは情報システムに対する要求であって、ソフトウェアの仕様に対する要求ではないことが重要である。この２つの要求の混同を避けるために、企業情報システム構築のプロセスを「情報システムサイクル」として規定し、ソフトウェアプロセスを情報システムサイクルに斜交する「ソフトウェアサイクル」として規定する。

情報システムサイクルの中で、情報システムに対する要求を「原要求」と呼ぶ成果物として規定し、それを建築学のパタンランゲージで記述することを提案する。パタンランゲージは、ソフトウェア工学において、知識の記述形式という側面のみに注目して、デザインパターンやソフトウェア・アーキテクチャ・パターンといった大きな成果を生んだ。しかし、パタンランゲージの本来の主張は、「時を越えた道」と呼ぶ継続的な活動である。その活動全体に着目すると、これは、原要求や設計制約の集積を、施主の組織学習として組み込んだ理想的な仕組みであることが分かる。

さらに、斜交する２つのプロセスの異なる価値観を調停する技術者としてのアーキテクトについて検討する。

---

*1 本論文は、児玉公信，水野忠則，「情報システム学的パタン･ランゲージの再発見」，情報処理学会研究報告，2007-SE-156，P49-56（2007）を発展させたものである。

## 1. はじめに

近年の企業情報システムは事業戦略と密接に関連している[1) 2)]が、システムの開発、保守、運用に膨大な費用と時間がかかるなど、抱える課題は少なくない。特に、事業環境の変化に俊敏に対応できないことは、事業の根幹を揺るがしかねない重大な問題である。たとえば、生産管理の領域で、個別受注（特注）による少量生産の比率が増えているのに、大量生産向けの生産手配システムを使い続けた結果、不良在庫が増加しているといった具合である。事業環境の変化に追随できない原因としては、変化の速度が速すぎて開発が追いつけない、あるいは開発費が回収できないということもあるが、企業側の情報システム担当者が問題を認識できない、ベテランが定年退職して業務全体を知っている人がいないので判断できないなどの周辺的な要因が積み重なって、惰性に流されている場合も少なくない。

不良在庫の例のように、企業情報システムの企画力低下や硬直化は、企業力を低下させることにつながる。情報システムにトラブルが起きた場合の社会的影響も大きくなり、責任主体としての対応能力も問われるようになっている。

こうした課題を解決するために、企業、特に情報システム部門が主体性を持って企業情報システムの企画、運営ができるように、整備計画管理（プログラムマネジメント）のためのフェーズ計画立案、概念データモデリング、ビジネスプロセスモデリング、要求記述の指導および教育が行われている[3)]。ただし、著者自身の経験から、多くの場合、顧客自身が形式的手法で要求文書を書くことは困難であると考えた方がよい。これと同様のことを、A. Davisもその著書「成功する要求仕様 失敗する要求仕様」[4)]のまえがきで述べている。顧客は形式的な要求記述に時間と労力を使うのではなく、より本質的で効果的なビジネスの仕組みを構想すべきである。

本稿では、このような状況認識の下に、情報システム部門の主体的な要求記述の活動と、それを支援するアーキテクトの役割について提案したい。まず第２章で、「時を越えた道」としての情報システムサイクルを定義し、第３章でパタンランゲージを導入する。次いで第４章でパタンランゲージを情報システムサイクルの原要求を述べる手段として用いることを提案し、第５章で情報システムサイクルと斜交するソフトウェアプロセスサイクルについて述べ、第６章でアーキテクチャと情報システムアーキテクトの役割について述べる。

## 2. 情報システムサイクル

現在の企業情報システムの構築は、都市計画に基づく建設のあり方に類似している[5)]。現代の都市計画は、既存のインフラストラクチャを生かしつつ、住民との合意を形成しながら、街区の再開発を漸進的に行う。現代の情報システムの構築も、既存のシステム基盤を生かしつつも、サブシステム単位で再構築を行い、既存のサブシステムとの連携も確保しなければならない。これは、長い時間をかけて少しずつ進めざるを得ない。このように、情報システムの構築方法について、長い歴史を持つ建築学、特に「まちづくり」の知見から学ぶことは多い。

### 2.1. 建築学に学ぶ

ローマ時代の建築家Vitruviusは、その著書[7)]の中で、アーキテクトは建築物に対して「用」、「美」、「強」という３つの「理」が保たれるよう働きかけるものだと述べている（図１）。ここで、「用」とは建築の用途、「美」とは形や外観だけでなく、住みやすさや使いやすさといった価値を指す。「強」とは構造のことであり、美を実現しながらも、重力に耐えうる構築物の建設方法を指す[6)]。つまり、アーキテクトはこの異なる３つの価値観を同時に実現させるよう設計しなければならない。*2

ただし、アーキテクトの仕事は設計だけではない。設計の実現を目指して、建築工事そのものにもかかわることもある。このため、アーキテクトは施主や施工者との調整を頻繁に行う。たとえば、日本のアーキテクトの第一人者である安藤忠雄の作品は、その意匠設計も優れているが、実際は、関係者との調整や現場との共同作業の上に成り立っている[8)]。

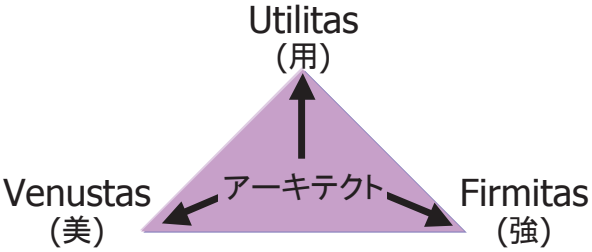



図１　アーキテクトの役割[6)]

*2 美を設計することを意匠設計と呼び，強を設計することを構造設計と呼ぶ。

### 2.1.1. Zachmanフレームワーク

情報システムの構築プロセスを建築プロセスと対応づけて規定した先行研究にZachman[9]がある。そこでは、アーキテクトが施主および施工者と対話することを通して、建築プロセスが進む様子が描かれている（表１）。

建築プロセスは、アーキテクトと施主との建物の用途、敷地面積などについてのやり取りに基づき、「バブルチャート*3」と呼ぶ粗い間取り図を手書きして基本的合意と信頼を得ることから始まる。受注後、アーキテクトは施主の視点で図面（アーキテクト図面）を書いて了解を取った後、施工者の視点で建物の詳細図面など（アーキテクト計画）を作成する。施工者は、これらの計画から、段取りや工事規制などの制約を加味して、施工方法（施工者計画*4）を作る。これをさらに現場ごとに細かい指示（現場計画）を作って渡し、建築作業が行われて完成する。

Zachmanは、情報システムの構築においても、上で述べたような建築プロセスと文書化（モデリング）の枠組みに沿った「規律あるアプローチ」によって実施すること、開発文書をデータ、プロセス、通信という３つの側面で記述することを提案した。その後、Sowa and Zachman[10]はこれを拡張して、文書記述の観点を5W1H*5とすることを提案した。

### 2.1.2. 施主、アーキテクトおよび施工者

情報システムの構築においても、施主、アーキテクト、施工者という立場が存在すると考えて、プロセスのあり方を整理してみよう。施主とは、企業全体の利益代表者であって、いわゆるCIOやその補佐役を含む組織全体を指すこともあれば、ある業務の責任者を指すこともある。アーキテクトは、設計者として、施主の構想に基づいてビジネスの

表１　Zachmanフレームワーク[9]

| ライフサイクル | 建築メタファ | ISドキュメント | データの記述 | プロセスの記述 | ネットワークの記述 |
|---|---|---|---|---|---|
| | バブルチャート | スコープと目標 | ビジネス上重要なエンティティの一覧 | 実施されるプロセスの一覧 | 事業拠点の一覧 |
| | アーキテクト図面（施主向けの表現） | ビジネスモデル | ER図 | 機能フロー図 | ロジスティックネットワーク |
| | アーキテクト計画（自分自身の表現） | 情報システムのモデル | データモデル | DFD | 分散システムアーキテクチャ |
| | 施工者計画（施工者自身の表現） | 技術のモデル | データ設計 | 構造チャート | システムアーキテクチャ |
| | 現場計画（文脈外の表現） | 詳細記述 | データベーススキーマ | プログラム | ネットワークアーキテクチャ |
| | | 機械語での記述 | | | |
| ↓ | 建物 | 製品（情報システム） | データ | 機能 | 通信 |

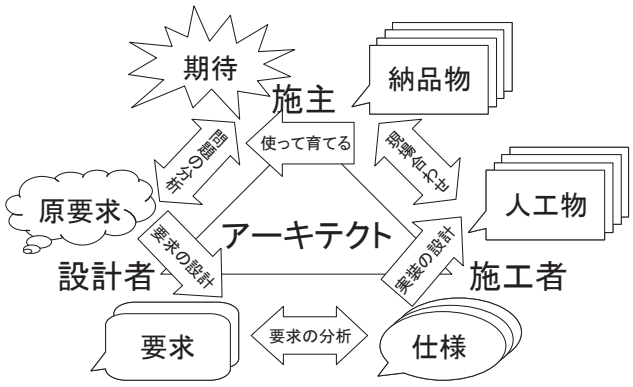



図２　情報システムサイクルモデル

*3 建築で言うバブルチャートは、部屋を書くために四角い。

*4 これを施工設計，または，特に最近では，生産計画と呼ぶ。

*5 もちろん、What, How, Where, Who, When, Whyのこと。

仕組みを設計する。施工者はそれをソフトウェアとして製造する。

ただし、アーキテクトの役割を一般的に規定するのは難しい。その役割は、主に設計者であるが、それ以上の役割が、施主、施工者、利用者、アーキテクトの協力者（たとえば弟子）、さらには周辺住民からなるステークホルダ間の力学的構造によって動的に創発される。

## 2.2. 情報システムサイクルモデル

上に述べたように、現代の企業情報システムは複合的な都市計画として扱わざるを得ない。これは、企業情報システムの構築も、継続的に時間をかけて行われる漸進的なプロセスであることを意味する。その結果、構築プロセスと成果物は、連綿と受け継がれた企業の「文化」を体現したものとなるだろう。企業文化は、民族特有の価値観の上に構築されるビジネスプロトコル*6と、その上に構築されるビジネストランザクション（業界のしきたり）、さらに各企業の歴史や特性が多層に織り込まれている[14]ので、非常に強固である。

以下で、Zachmanがとらえた情報システムの構築プロセスを、継続的に循環するサイクルとして再定義する。

なお、Zachmanフレームワークの各文書レベルと、情報システムサイクルの各中間成果物との対応は次のとおりである。「バブルチャート」は原要求に、「アーキテクト図面」と「アーキテクチャ計画」は要求に、「施工者計画」は仕様に、「建物」は製品に対応する。なお、「現場計画」はソフトウェアプロセス（後述）の内部で使われるので、情報システムサイクル上で対応するものは現れない。

### 2.2.1. 成果物の変換プロセス

情報システムの構築に参加する施主、設計者（としてのアーキテクト）、施工者の関係と、その間に流れる中間成果物を図２のようにとらえる。中間成果物は参加者それぞれが果たす役割によって次々と変換されていく。これを「情報システムサイクル」モデルと呼ぶことにする。

情報システムサイクルの開始トリガは、施主による「期待」の表明である。これは目標の方向付けである。その期待に対してどのように事業を行うか、ステークホルダ間で合意を取って、具体的な目標値および目標状態を非形式的に記述したものを「原要求」と呼ぶことにする。原要求は設計者によって「要求」と呼ぶ形式的な記述に変換され、その内容について施主と合意する。要求記述は、施工者が設計者と打ち合わせて製造「仕様」に翻訳し、その内容について設計者と合意する。これをプログラムとデータ、マニュアルなどの「人工物」に変換し、検査を経たのちに、施主に引き渡す。その際、軽微な修正が行われることがある。これを「現場合わせ」と呼ぶ。施主は現場合わせされた人工物を「納品物」として受け取り、実際に使うことで「効果」を得る。さらに効果が発揮されるように改良を行うこともある。これを「使って育てる*7」と呼ぶ施主もいる。

一般に、企業情報システムの構築は、部分的かつ段階的に行われるため、１回のサイクルで施主の期待が満足されることは稀であって、このサイクルは継続的に繰り返される。また、この間に施主の事業環境が変化することもあるので、このサイクルは事業が続く限り回り続ける。

### 2.2.2. 「要求」と呼ばれる成果物

ここで、「要求」と呼ぶ成果物が設計者によって「設計」されることに違和感があるかも知れない。しかし、現在のソフトウェア工学では、ソフトウェアがどうあるべきかの記述を「要求」あるいは「要件」と呼んでいる*8。一般に施主にとっての"要求"は、ビジネスがどうあるべきかの表明であり、本稿でいう「原要求」に相当する。施工者にとっての「要求」は、むしろ「原要求」の概念レベルでの実現方法を指していることが多い。

さすがに、施主の「要求」と施工者の「要求」とのギャップが問題だと感じる実践者もいて、要求工学[15]や要求開発[16][17]というスローガンを掲げているが、それらの多くも要求の意味を下流にずらしている。

ともあれ、「要求」という語に問題はあるが、別な語を当てる方がかえって混乱を招くと思うので、本稿ではこのまま用いる。

*6 たとえば、請求書とinvoiceの違いなどを指す。

*7 この言葉は情報システム学の立場から非常に含蓄がある。育てられるのは、多くはソフトウェアではなくて利用者のほうだからである。こうした相互作用の存在がシステム的と言える。

*8 英語ではともにrequirementsであり、区別されない。

### 2.2.3. ソフトウェアプロセスの位置づけ

ソフトウェアを分析し、製造し、検査して納入するまでのプロセスをソフトウェアプロセスあるいはソフトウェアライフサイクルプロセスという。この一連のプロセスを、ここでは施工者が行う情報システムサイクルの内部プロセスと見なす。これは、要求分析、ソフトウェア設計、ソフトウェア製造というステップを踏むが、情報システムサイクル上ではそれを明示しない。このとらえ方については第５章で述べる。

### 2.2.4. 「情報システムサイクル」の意義

この情報システムサイクルは、新たにプロセスを設計したものではなく、情報システムサイクルとソフトウェアプロセスの区別がなかった時代のソフトウェアプロセス、たとえばBoehmのスパイラルモデル[18]のとらえ直しに過ぎない。しかし、この情報システムサイクルを認識すること、特に原要求を明確に記述することによって、施主の主体的な活動を規定できる。

## 2.3. 施主による「期待」の表明

期待は、施主が求めるビジネス目標の方向付けであり、これには一切の設計を含めない。期待は、施主自身の言葉として述べられるべきである。このため、たとえば、物語として語られることも多い。これがうまく語られることで、施主チームのトラストビルディング*9が図られる。この好例が、日産自動車のスカイラインR30モデルの開発リーダ、桜井真一郎氏が語った「戦場ヶ原の稲妻[11]」と呼ばれる物語である*10。そこでは、どのような人がどのように使うかのイメージが、ロマンチックな物語を通して語られた。

## 2.4. 原要求を作る

要求記述の前に原要求を規定する。その理由は、施主自身の言葉で要求を述べることによって、施主のコミットメントを確保し、施主が継続的活動の主体であることを強調したいためである。また、原要求を施主の成果物として、その内容の妥当性を担保しておきたいこともある。アーキテクトは、施主のコミットメントを引き出すと同時に、原要求の欠陥が除去されるように働きかける。

### 2.4.1. 原要求の位置づけ

原要求の記述レベルは、「期待」と「要求」記述との間のどこかにある。どこにあるかは企業のビジネスの特質や情報システムの発達段階によって異なる。たとえば、システムの応答性がビジネスを左右するならば、性能や安全性への要求が原要求に含まれるだろうし、情報リテラシが不十分な企業では、期待を機能要求として言い換えただけの原要求もありうる。たとえば、「利益を最大化する機能」というようなものである。もちろん、このままでは後工程で問題が起きるので、アーキテクトは、原要求の内容が適正なものとなるよう、施主に働きかける必要がある。いずれの場合も設計者がそこから設計できる何らかの情報が含まれていなければならない。

原要求には、企業情報システム全体に対するものと、その部分である個別の業務システムに対するものがある。初期的には、これらは渾然一体となっているが、情報システムサイクルが回るに連れて変化し、分化していく。

### 2.4.2. これまでの原要求導出の方法論

これまでに知られている方法論のうち、システムに対する働きかけの循環的な修正を強調しているものは、P. Checklandらが提案するソフトシステム方法論（SSM）[12]や手島の手法[13]である。これらは、働きかけに対するシステムの変化を観察することによって概念モデルを変更していくことをその方法論に含めている。

このほかに、大手システムベンダから提供される問題分析手法がある。これらは、問題－原因ネットワークを書いたり、カウンセリングの手法やKJ法と類似の手法を用いたりして、巧みに合意形成に誘導する。この結果として機能要求が得られれば、それを原要求とすることができる。ただし、これらの手法は１回での完結を期待し、プロセスの循環性を念頭に置いていないものもある。情報システムの対象は、システムアプローチでいうソフトシステム[12]である。ソフトシステムにおける問題解決では、原理的に正しいと言える解決策はない。合意に基づいて解決策を実

＊9 チームへの忠誠心を高めるための手法。もちろん，桜井氏はそれを意識していないと思う。

＊10 それは、恋人が待つホテルに向かって雨の中を急ぐ車が、暗闇の中で稲妻によって一瞬浮かび上がるイメージであった。

施し、システムの反応を見て、次の解決策を探るという循環的なアプローチが必須となる。

### 2.4.3. 原要求に書かれるもの

SSMでは、対象システムの概念モデル[19) 20)]を書いて、これと現実活動との差を効果的に解消するような改善策を立てる。これが原要求として書くべき内容に当たる。

ほかに、IEEE Std 1362-1998[22)]の「構想書」の内容が参考になる。これには、現状の記述から始まって、変更の必要性、新システムの構想、運用のシナリオ、新システムのインパクト、効果、注意事項などを書く。ただし、この構想書のテンプレートで原要求を書くときには、ソフトウェアの仕様記述部分は不要である。原要求は、ソフトウェアに対する要求ではないからである。

### 2.4.4. 原要求のテスト

記述された原要求について、その内容の正しさと、施主チーム内の合意の存在を確認し、施主のコミットメントを得るために、原要求記述が完成した時点で、直ちにそのテストを行う。これは、「V字型ソフトウェアプロセスモデル」の後半でのユーザ受入テストを前倒しで実施することを言っているのではない。これはソフトウェアのテストであって、原要求のテストではない。

原要求のテストは、施主自身が原要求の欠陥を除去する活動とする。そのために、ソフトウェアに関する記述を排除し、施主自身やステークホルダが分かりやすい形式、たとえば、ユーザシナリオ、デモビデオ、絵コンテなどを用いて、「ウォークスルー」できるようにしておく。

### 2.5. 設計

ここでいう設計とは、原要求を満足する具体的な方法を考案し、「要求」に変換する作業のことである。要求は、建築の領域でいう意匠設計と構造設計の結果に相当する。この変換作業は、設計者の高度な発明あるいは創作であり、他者の干渉を受けない。設計者がレビューを要請することはある。

企業情報システムの要求の記述は、業務システムのスケッチとして、全体構想、概念データモデル、ビジネスプロセスモデル、概念ユースケースなどからなる。これは施工者に対するインプットとなり、施工上の都合や、より安価で短工期の製造方法などのフィードバックを受けて、多少修正される。

要求も、原要求のテストと同様、文書の完成時にテストされるべきである。これは、施主に対する、一般的説明、一部のプロトタイプ、デモによって行われる。施工者は、この要求に基づいて製造にかかる費用を見積もる。

要求記述のテンプレートとして、IEEE Std 830-1998[23)]やVolereアプローチ[24)]が参考になる。ただし前者は、ソフトウェアの要求のテンプレートであり、本稿でいう情報システムの要求記述として書くときは、製造仕様に関する記述を排除する。

### 2.6. 製造から引き渡しまで

要求が仕様に変換され、人工物が製造され、現場合わせされるまでの過程は従来の考え方と変わらない。この過程は、Zachmanでも見たように、複数の下位作業に分割されて、同時並行的に進められることもある。

また、アジャイルと総称されるソフトウェアプロセス[21) 32)]では、この過程を頻繁かつ高速に繰り返す。この手法は、施主の全面的な協力と、アーキテクトが設計者と施工者を兼ねることで実現される。

## 3. パタンランゲージの再発見

原要求の記述および要求への変換について、建築学のパタンランゲージという理論を参考にして考えてみる。

パタンランゲージとは、1970年代、カリフォルニア大学の建築学の教授であったChristopher Alexanderが主張した、街や家の建築計画と設計法のアイデアである。彼は、得も言われぬ心地よさ*11を持つ町並みや建築物は、古くから技術者に伝えられてきたが今は忘れ去られた「パタン」を持っていると考えた。それらを明らかにし、適切に組み合わせて使うことによって良いまちづくりができるとして、いくつかの実験的な試みを行った*12。

*11 これを無名の質（QWAN; Quality Without A Name）と呼ぶ。

*12 ただし、これらの実験が成功したという評価は必ずしも得られていない。これには多くの事情があったと思われるが、弟子たちは多くのプロジェクトで実践し、成果を得ている。

このパタンランゲージの考え方がソフトウェア工学に与えた影響は大きい。その成果である「デザインパターン」[30]やそれに続く多くのパターンカタログ、そしてそれらを生み出す原動力となったPLoPの活動[31]が、ソフトウェア製造における品質の向上に貢献した。

しかし、これまでの情報技術におけるパタンランゲージの活用は、ノウハウ記述としてのパタンの側面だけに限られていた。本来のパタンランゲージとは、「時を越えた建築の道」[25]を行く「活動」であるべきである。これについて以下で述べる。

### 3.1. 活動としてのパタンランゲージ

数十年あるいは数百年に渡って続く建築の営みを、マスタプラン（基本計画書）によって制御することはできない。しかし、得も言われぬ心地よさを実現するためには、何らかのガイドが必要である。その意味で、パタンは緩い計画書であり、建築チームに対する過去からのアドバイスである。これを受け取り、良い建築を残し、有用なアドバイスを未来に引き継いでいく活動がパタンランゲージである。

#### 3.1.1. ルール

建築プロジェクト開始の前に、パタン活動を支える原則的なルールが制定される。ルールとは、プロジェクトの遂行におけるさまざまな活動の原則を示したものである。

「オレゴン大学の実験」[27]では、①有機的秩序、②参加、③漸進的成長、④パタン、⑤診断、⑥調整からなる６つのルールが掲げられた。「パタンランゲージによる住宅の建設」[28]では、①アーキテクトビルダ、②ビルダーズヤード、③共有地の共同設計、④個々の住宅のレイアウト、⑤一歩一歩の施工、⑥コストコントロール、⑦プロセスの人間的なリズムの７つのルールとなっている。このほかのAlexanderのプロジェクトでも、それぞれの目的に合わせてルールが定められている。

#### 3.1.2. 計画評議会

「オレゴン大学の実験」では、ルールに基づいて、パタンランゲージの制定および改訂を諮る組織が設けられた。これは、施主（大学）、利用者（学生）、アーキテクト*13の三者からなっていた。施工者は含まれない。これは、決定の権限を持つように、委員会ではなく評議会（board）とされた。ほかのAlexanderのプロジェクトで、計画評議会に相当する組織を設けたことの記述はないが、どれも施主や利用者を巻き込んだ活動になっている。状況に合わせて適切な体制が作られるようである。

#### 3.1.3. パタンフォーマット

パタンの記述様式は独特である。まず、パタンに名前を付ける。たとえば「４階建ての制限」、「老人の離れ」、「貸せる部屋」というようなキーワードである。次いで、そのパタンについて、背景、問題、解決策、フォース（force）という４つの節を設けて記述していく。フォースには、注意事項として、他のパタンとの関係、組み合わせの可否などを書く。

これはアーキテクチャの断片の記述である。特に、最後の節、フォース*14が重要である。フォースとは「力」という意味であるが、建築の世界では主に「引力」を意味する[6]。つまり、本来のアーキテクチャとは引力を克服する方法だったのである。そして、アーキテクチャの断片を組み合わせて構造を実現する。

情報システムのアーキテクチャが克服すべき「力」とは、複雑性であろうか。著者は、まだしっかりとは分かっていない。

### 3.2. 計画ランゲージ

Alexanderの著書「パタンランゲージ*15」[26]には、パタンランゲージの使い方が書かれている。まず、253のパタンの中から、建築しようとするテーマに沿って、語り始めのパタンを探して、前後のパタンを参考にして、最初のパタンを決める。次に、後ろに向かって有用と思われるパタンをすべて抜き出す。これに対して、必要に応じてパタンを変更し、追加する。こうしてできたパタンの並びを「計画ランゲージ」と呼ぶ。これはパタンを語彙とする１つの詩あるいは物語であり、それを語る順序が設計の順序

*13 このアーキテクトとはAlexander自身であった。

*14 Alexander自身，A Pattern Languageの中ではこの言葉を使っていない。

*15 原書の標題は"A Pattern Language"で，たくさんありうるパタンランゲージのうちの１例であることを強調している。

に対応する。

### 3.2.1. ありありと語る

計画ランゲージの例として、Alexanderが実際に手がけた日本の盈進学園東野高校の例[34]を示す（図３）。ここでは、「オレゴン大学の実験」に倣って、経営者、利用者である教師、学生、アーキテクトからなる計画評議会が設けられ、計画ランゲージが作られた。ここで、下線部分がパタンであり、これに対応するパタン集が別に作られる。このパタン集は入手できていない。

図３の記述を読むと、学校の建物群の中をまさに歩いている（walk-through）かのように、これから出来上がる学校のイメージがありありと浮かぶ。これは、校舎の設計ではなく、学校生活の設計であると言えるだろう。そして、これは、施主にとってテスト可能な記述でもある。

パタンは、施主、アーキテクト、施工者の間で通用する技術語彙である。たとえば、「玄関道」といえば、３者が直ちに理解する。もし誤解があれば修正する。

### 3.2.2. 現場でのイメージアップ

このほかに、Alexanderは建設現場に行って棒を立てて間取りを決めたり、紙でドアのイメージを書いて貼ったりして、現場で原要求を具体化する手段を取っている。

### 3.2.3. デザインコード

パタンランゲージおよび計画ランゲージは設計者の自由度を制約するデザインコードの意味を持つ。自由度は制約されても、設計者が変換作業で用いる作業空間が狭められることで、設計効率は上がると期待される。また、異なる設計者が設計しても、景観上の秩序[33]は守られる。

## 4. 情報システムパタンランゲージの構想

このパタンランゲージの考え方に基づいて、情報システムサイクルにおける原要求の導出方法と記述方法について私案を述べる。なお、付録に小さな事例を載せた。

2-3 玄関道
– 正門より内境界に向けて「玄関道」がある。玄関道の両側には壁か樹木が並び非常に静かである。
2-5 中央広場
– 第2の門の内側には、中央広場がある。この広場は大講堂とともに構成され、講堂の正面は庭に向かっている。
2-6 大切な中心
– 中央広場の先に、第3の門を通り抜けると高校と大学の最も大切な中心がある。ここは幾重もの層を通り抜けて到達できる場であり、そこには静かさがある。
2-8 大切な中心から伸びる腕
– この中心から枝分かれして、個別・分離した大学・高校部分が始まる。
2-9 田の字センター
– この大切な中心は、心臓部であり、高校と大学の交差するところである。そこは2つの道が交差する十字形になるので、我々は「田の字センター」と名づける。
4-6 田の字センターの繁華地区（中枢部）
– 田の字センターの半分は、全学的に使用され、より繁華である。その場所は、学生が毎日動いている場所である。そこはまた、卒業生が集まる広場でもある。
4-7 田の字センターの静かな地域
– 田の字センターの他半分は、より神秘的である。大学回廊に向かっている部分で、たぶん学生会館の裏手となる。この部分は、入り口からあるいは柱の間やアーケードを通して一瞥することができる。しかし、そこは静かで高校生には行くことができない。その部分を一瞥するたびに、高校生たちは、そこで受ける教育に対する憧憬を抱くのである。

図３　計画ランゲージの例

### 4.1. 情報システム版パタンランゲージ

情報システムサイクルに、パタンランゲージを導入した場合の、作業の流れをEriksson and Penker[35] のプロセス図を使って図４に示す。

#### 4.1.1. 情報システム版計画評議会

企業のパタンランゲージを作り、維持するのは計画評議会に相当する組織である。これが図４の上半分のサイクルである。個別の業務システムの開発では、図４の下半分に示すように、既存のパタンランゲージを、施主は原要求を述べるときの語彙として、設計者はデザインコードとして、施工者は施工基準として利用する。

利用者は実装物を実際に使用した上で、設計の良否、実装の良否を評価し、計画評議会はその評価を受けて、必要に応じてパタンランゲージに修正を加える。これによって、特に、業務領域の原則、業界の掟など、明示されにくい知識も蓄積され、継続的に企業システム全体の質が改良されていくと期待される。

施主チームが、個別のソフトウェア実装案件のリリースタイミングを調整する。これが企業情報システムの成長速度を左右する。

#### 4.1.2. マスタパタンランゲージの作成

各企業がパタンランゲージの活動を始める前に、Alexanderがやったように、参考となるパタンランゲージがあらかじめそろっているとよい。これは、過去の要求記述に相当する成果物の観察から行う。

要求記述や提案書の中に、施主とのインタビュー集があれば、そこから事実の言及、仕組みに対する言及を除いたものを「期待」として取り出す。これから「原要求」を生成し、技術用語として取り上げるものをマークする。マークしたものと実際に記述された「要求」の機能とを対応づけて、それをパタンフォーマットで記述し直す。

こうして集まったパタンランゲージの候補を、対象の大きいものから小さいものまで順に並べて、重複があればそれを調整し、ギャップがあればそれを埋めるようなパタンを追加する。全体として社会、組織、業務となるように体系化する。これを複数の構築計画から集めることで、Alexanderの「パタンランゲージ」と同様に利用できるようになる。これをマスタパタンランゲージと呼ぶ。

#### 4.1.3. 原要求の記述

原要求の記述の仕方は、まさに計画ランゲージの作り方である。「期待」が与えられていれば、それぞれについて実現値や目標状態を想定して合意を取る。それに基づいて、パタンランゲージの中から最初のパタンを探して、必要なパタンを抜き出す。設計につながらないキーワードがあれば、その分のパタンを新たに書き起こす。こうしてできた計画ランゲージに対して、施主自身がありありとイメージを浮かべられるようになるまで修正する。

ただし、この時点では新しいパタンは解決の方向が示されただけの未完成の状態にある。これは、実際に設計され、実装され、運用され、評価されてはじめて、デザインコード、施工基準となる再利用可能なパタンとして完成する。計画評議会はこうしたパタンの状態を把握しておき、有効とされたパタンをマスタパタンランゲージに加える。これは過去のパタンも同様である。

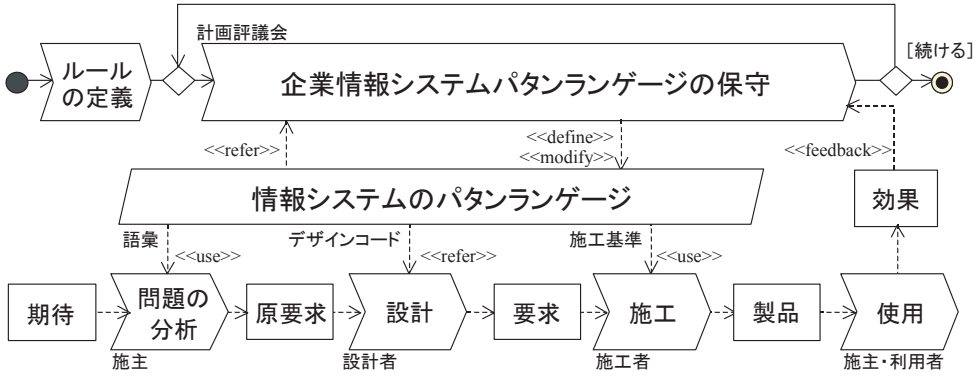



図４　パタンランゲージによる情報システムサイクル

#### 4.1.4. 要求管理

原要求がパタンを使って語られ、パタンの実装が要求として記述されることで、期待から要求までのトレーサビリティが明らかになる。ここまでのプロセスは並行的に実施されてもいいし、前の工程に逆流して成果物を修正してもかまわない。

## 5. 情報システムサイクルとソフトウェアプロセス

情報システムサイクルについて整理し、これまで曖昧になりがちだった原要求の扱いをパタンランゲージという活動を通して明確化する試みについて述べた。これと対置すべきソフトウェアプロセスの位置づけについても、簡単に整理しておく。

### 5.1. ソフトウェアサイクル

上でも述べたように、ソフトウェアプロセスを情報システムサイクルの内部プロセスとして扱う。かつてはこれが施主の内部要員によって行われることが多かったが、現在ではソフトウェアベンダの作業であることが多い。

施主の立場から見ると、このプロセスは計画ごとに、通常１回しか行われないが、ソフトウェアベンダは類似の作業を他の施主に対して繰り返して行ってきており、その知見の蓄積もある。ここにはベンダ側の論理に基づく自己フィードバックループがある。これをソフトウェアサイクルと呼ぶことにする。

このようなサイクルは、現在、ソフトウェアプロダクトラインアプローチ[36]として議論されている。これは、ソフトウェアをコアアセット（共通部品）とバリアント（個別部品）に分けて、組み立てていく考え方である。共通か個別かの判定や、部品の進化は、同様なアプリケーションを何度か繰り返さなければ得られない。このイメージを図５のプロセス図に示す。

### 5.2. 交差する２つのサイクル

施主の情報システムサイクルと施工者のソフトウェアサイクルとは交差している（図６）と考える。ここで、施主はさまざまなプログラムを継続的に実施し、施工者は、異なる施主に向けて同様なソフトウェアを継続的に製造している。この交点で、要求の受け取り（施工者の立場からは「要件定義」と呼ばれる）から製品の受け渡しまでが行われる。施工者はこれをプロジェクトと呼ぶ。

この見方を通して強調したい点は、施工者作業の上流に、要求、原要求、期待は存在しないことである。ソフトウェアプロセスでは、与えられた「要件」に基づいて「速く、安く、バグのない」製品を作る*16ことがその価値観である。だから、ソフトウェア工学に立脚する要求工学では、「要件」が誤解されないように形式的に書かれることを強調する。情報システム学の価値観は、「効率的、効果的」にビジネスを支援する*17ことにあるので、これら二つの異なる価値観が、この交点での要求解釈のすれ違いを産む。たとえば、施主が在庫を減らしたいと考えているとき、施工者は在庫管理の機能を提供するといった具合である。状況

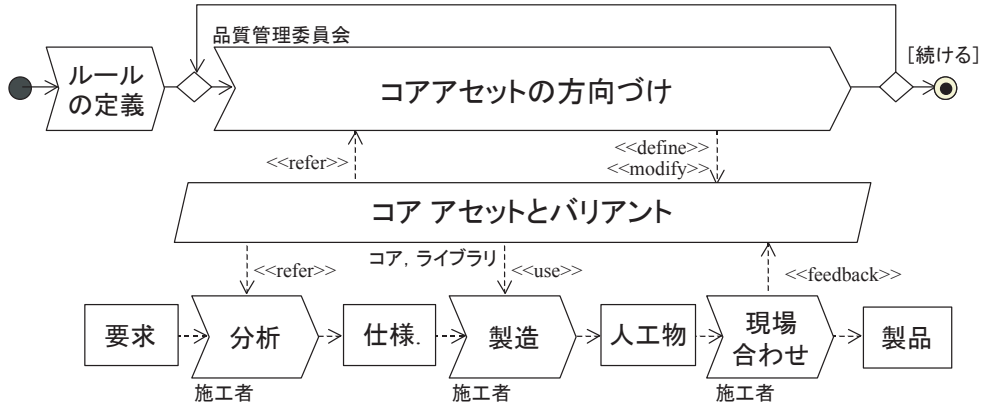



図５　プロダクトラインを組み込んだソフトウェアサイクル

*16 つまり，QCD（Quality, Cost, Delivery）で，ものづくりの価値観である。

*17 ソフトシステムズ方法論[20]では，Efficacious, Efficiency, Effectiveとされる。ただし，その計測は難しい。

にもよるが、本来は、実在庫に引き当てるのではなく、予定在庫に引き当てるように生産計画を立てられる機能を提供すべきである。と同時に、現場の生産実行作業の精度を高める必要がある。後者はソフトウェアの問題ではないが、情報システムの問題である。

施主、施工者、設計者の異なる価値観を同時に実現するのがアーキテクトの役割であろう。この点に関して、次章で問題を提起したい。

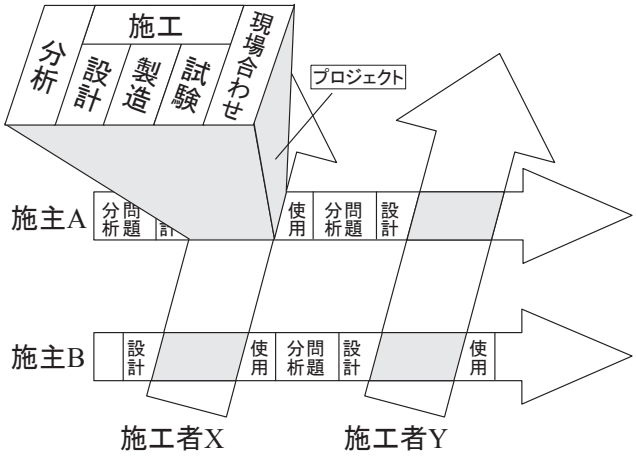



図６　交差する情報システムサイクルとソフトウェアサイクル

## 6. アーキテクトの位置づけ

最近、情報処理業界において「アーキテクト」という語がよく使われるようになった。しかし、その意味が確立して通用する前に、さまざまな思惑で使われていて混乱を生んでいるのが実情である。上で述べた情報システムサイクルとソフトウェアサイクルという視点から、「アーキテクト」の役割について、あらためて検討する。

### 6.1. 資格としてのITアーキテクト

情報処理推進機構（IPA）のITスキル標準センターが策定したITスキル標準（ITSS）[37]では、ITアーキテクトなる職種を設定して、その中の専門分野として、アプリケーションアーキテクト*18の達成指標を記述している。そこでは、「ビジネスおよびIT上の課題を分析し、ソリューションを構成する情報システム化要件として再構成」し、特に「アプリケーションアーキテクチャを設計する」ものとしている。これは、ビジネスとITで両立しない課題を解決するよう設計するということで、本稿が主張する情報システムのアーキテクトに相当する。しかし、後半の、アーキテクチャを設計する者がアーキテクトであるというのには同意できない。その理由を6.2節で述べる。

一方、経産省が主催する産業構造審議会情報経済分科会の報告書には、このITSSと、IPAのソフトウェアエンジニアリングセンター（SEC）が策定した組み込みスキル標準（ETSS）、経産省が策定した情報システムユーザスキル標準（UISS）とを、「共通キャリア・スキルフレームワーク」として一本化する方向が示されている[38]。その案では、３つの人材類型を設定し、それぞれに具体的な職種例を、そして要求されるスキルセットを例示している。本稿で議論しようとしているアーキテクトは、ソリューション系人材に属し、「システムアーキテクト」という職種が例示されている。このシステムアーキテクトは、「IT戦略を受け、ソリューションを構築する*19」ものとされる。ただし、このほかに、ITSSからのITアーキテクトと、UISSからのISアーキテクトという名称も残っており、今後、整理されることになる。

### 6.2. アーキテクチャとアーキテクト

ITアーキテクチャの定義もさまざまだが、ここではIEEE Std 1471-2000の定義を挙げる。これによると、「コンポーネントおよびコンポーネントと他のコンポーネントや環境との関係で具現化されるシステムの基本構造、およびこれらの設計や進化を統括する原理」[39]となっている。ここでいう「システム」とは、一般システム理論[40]でいうシステムであると理解しなければならない。すなわち、観察対象を「交互作用する多くの要素を持つ全体」ととらえたものである。したがって、さまざまなレベルでシステムが存在し、それごとにアーキテクチャが存在する。たとえば、オペレーティングシステムには、タスク管理、仮想記憶管理、データ管理のアーキテクチャがあり、生産管理シ

*18 ほかに、インテグレーションアーキテクト、インフラストラクチャアーキテクトの２種も挙げているが、本稿では論じない。

*19 この後ろは「又は組込製品開発に必要となる要件を定義し、それを実現するためのアーキテクチャを設計する」となっている。

ステムには、技術データ管理、計画管理、現場制御のアーキテクチャがある。

これらは、設計してできるものというよりも、「時を越えた」繰り返しの活動の中で有効なものとして評価され、生き残った技術体系というべきである。実際、建築のアーキテクチャ[6]は、長い歴史の中で、自然選択の原理に基づいて洗練されながら、連綿と受け継がれてきた。ソフトウェアアーキテクチャも同様であり、Shaw and Garlan[41]によると、「それは目で見てわかるものではなく、明確に記述されるものでもなく、むしろ設計からの抽象として時間をかけて創発されるものだ」としている。

アーキテクトが設計しようとするのはシステムであって、アーキテクチャではない。多くの場合、既存のアーキテクチャをうまく組み合わせることで、施主の目的を達成するようなシステムを設計しようとする。

アーキテクトは、こうしたアーキテクチャの選択がシステムの基本構造として最適であったかを自己評価して、自らのアーキテクチャスタイルとして確立していく。このために、アーキテクトは、学会や技術士会、プロフェッショナルコミュニティなどの活動を通して、さまざまなアーキテクチャスタイルを知り、評価の目を養う。これが本稿での情報システムのアーキテクト像である。アーキテクチャスタイルを表現する有力な手段の１つとして、上で述べたパタンランゲージが使える可能性がある。

## 7. おわりに

最後に、日本の情報処理産業のあり方について自由でオープンな議論を期待して、個人的な意見を述べさせていただく。

企業情報システムは、1980年代の前半までは、各企業が自前で作るのが普通であった。1980年代の後半からは、情報システムが経営戦略の一翼を担うようになって、巨大化、複雑化し、プロフェッショナルが代わって作る時代になった。そして多くの企業が情報子会社を作ることになった。エクサもこうして20年前に誕生した。1980年代はまさにソフトウェア工学が花開いた時代[42]であり、ソフトウェアを作ることこそが自分たちの価値であると思っていた。

Zachmanが情報システムのフレームワークを提案したのも、ちょうど20年前のこの年である。そして現在、エクサを含め多くのSI会社は岐路に立っていると思う。ソフトウェアの製造で生きるのか、情報システムの構築で生きるのか、あるいはその両方で生きるのか。両方で生きるとすれば、異なる価値観の技術者をどのように育て分けるのか。

ソフトウェア製造は、個別受注型の「ものづくり」ビジネスに似ている。これは、品質、工期、コストをたえず最適化していくダイナミックな活動である。ものづくりの活動は、国際標準IEC 62264-1[43]によれば、要求変更管理、計画立案・手配、生産コントロール、部品・コンポーネント管理、調達管理、品質保証、成果物管理、納品物管理、設備管理、マーケティング、研究開発などからなる。ソフトウェアの製造で生きるためには、「ものづくり」の活動に対応したこれら製造技術の確立、特に製造リードタイムの短縮が不可欠となる。プロジェクト管理もこの製造技術の一部である。

一方、情報システムの構築は、施主のビジネスに貢献していく活動である。そのために、ビジネスにとって効果的（用）であること、使いやすいこと（美）、変化に強い構造を持っていること（強）という３つの価値観を同時に満足するよう導く情報システムアーキテクトの存在が不可欠である。これを本稿では「まちづくり」になぞらえ、施主とともに「時を越えた道」を行くことであるとした。

情報システムのアーキテクトがどのように行動すべきかについては、国際標準どころか合意された方法論がないアート（匠の技）の領域である。したがって、アーキテクトの育成は、座学や単純なOJTでは達成できない。もしエクサがここで生きていこうとするならば、情報システムのためのアーキテクトとそれを補佐する優秀な設計者の育成が欠かせない。両者の育成方法は、こうした問題意識の上に、これまでのキャリア開発計画とは異なるものになるだろう。ここに、情報システム版パタンランゲージの活動が寄与できるのではないかと考える。

### 参考文献


1) 経営システム研究会（編）.「NTTドコモ リアルタイムマネジメントへの挑戦」, 日刊工業新聞社(2004)
2) 経営情報学会システム統合特設研究部会（編）.「成功に導くシステム統合の論点」, 日科技連(2005)
3) 手島歩三.「悩み深い情報システム部門の再生策」, 日経コンピュータ, 2002/08/26号, 日経BP, P 172-177
4) Davis, A. M.著. 萩本順三ほか訳.「成功する要求仕様 失敗する要求仕様」, 日経BP社 (2006)
5) 南波幸雄.「企業情報システムにおけるシステム統合と都市計画アプローチ」, in「システム統合の論点」, 日科技連, P

79-92 (2005)
6) O'Gorman, J. F., "ABC of Architecture, " PENN (1998)
7) Vitruvius著. 森田慶一訳,「ウィトルーウィウス建築書」, 東海大学出版会(1979)
8) 平松 剛.「光の教会―安藤忠雄の現場」, 建築資料研究社 (2000)
9) Zachman, J. A., "A Framework for information systems architecture," IBM SYSTEMS JOURNAL, Vol. 26(3), 1987. Reprinted Vol. 38(2, 3) (1999)
10) Sowa, J. F. and Zachman, J. A., "Extending and formalizing the framework for information systems architecture," IBM SYSTEMS JOURNAL, Vol. 31(3) (1992)
11) http://www.motorfan.jp/modules/xf6section/article-17.html
12) Checkland, P., "Systems Thinking, Systems Practice," John Wiley & Sons (1981). 高原康彦ほか監訳.「新しいシステムアプローチ」, オーム社 (1985)
13) 手島歩三ほか.「概念データモデル設計によるソフトウェアのダウンサイジング」, 日本能率協会マネジメントセンター (1994)
14) Chroust, G., "Software Like a Courteous Butler: Issue of Localization under Cultural Diversity," Proceedings of the 51st Annual Meeting of the International Society for the Systems Sciences, 2007-747 (2007)
15) Loucopoulos, P. and Karakostas, V.著 富野 壽訳.「要求定義工学入門」, 構造計画研究所 (1997)
16) 山岸耕二ほか.「要求開発~価値ある要求を導き出すプロセスとモデリング」, 日経BP社 (2006)
17) Robertson, S. and Robertson, J., "Requirements-Led Project Management: Discovering David's Slingshot" Pearson Education, (2007) 河野正幸訳.「ソフトウェアの要求『発明』学」, 日経BP社 (2007)
18) Boehm, B. W., "A Spiral Model of Software Development and Enhancement," COMPUTER, IEEE, pp. 61-72 (1988)
19) Wilson, B.著,「システム仕様の分析学―ソフトシステム方法論」, 共立出版 (1996)
20) 根来龍之.「SSM入門テキスト」(未公刊), 1994年6月
21) Poppendieck, M. and Poppendieck, T. 著, 平鍋健児訳.「リーンソフトウェア開発~アジャイル開発を実践する22の方法~」, 日経BP (2004)
22) IEEE Std 1632, "Guide for Information Technology System Definition -- Concept of Operation Document," IEEE (1998)
23) IEEE Std 830, "Recommended Practice for Software Requirements Specifications--Description," IEEE (1998)
24) Robertson, S. and Robertson, J. "Mastering the Requirements Process, 2nd Ed.," Addison-Wesley (1999)
25) Alexander, C., "The Timeless Way of Building," Oxford Univ. Press, (1979). 平田翰那訳.「時を越えた建築の道」, 鹿島出版(1993)
26) Alexander, C. et al. "A Pattern Language," Oxford Univ. Press (1977). 平田翰那訳.「パタン・ランゲージ」, 鹿島出版(1984)
27) Alexander, C. "The Oregon Experiment," Oxford Univ. Press (1975). 宮本雅明訳.「オレゴン大学の実験」, 鹿島出版 (1977)
28) Alexander, C. et al. "The Production of House," Oxford Univ. Press (1985). 中埜博監訳.「パタンランゲージによる住宅の建設」, 鹿島出版 (1991)
29) Alexander, C. "A New Theory of Urban Design," Oxford Univ. Press (1987). 難波和彦訳.「まちづくりの新しい理論」, 鹿島出版 (1989)
30) Gamma, E., et al. "Design Patterns, Reading," MA, Addison-Wesley (1995). 本位田真一ほか監訳.「デザインパターン(改訂版)」, ソフトバンクパブリッシング (1999)
31) Hillside Group. URL http://hillside.net/
32) Beck, K., et al. "Extreme Programming Explained: Embrace Change," Addison-Wesley (1999)
33) 真鶴町.「美の基準―デザインコード」、真鶴町まちづくり条例 (1993)
34) 羽生田栄一. Personal communication (2007)
35) Eriksson, H. and Penker, M.著. 本位田真一ほか監訳.「UMLによるビジネスモデリング」, ソフトバンククリエイティブ (2002)
36) Sugumaran, V., Park, S, and Kang, K. C. "Software Product Line Engineering," CACM, Vol. 49 (12), P29-32 (2006)
37) 情報処理推進機構,「職種の概要と達成度指標」, ITスキル標準V2 2006 (2006)
38) 産業構造審議会情報経済分科会情報サービス・ソフトウェア小委員会,「高度IT人材の育成をめざして」, 人材育成ワーキンググループ報告書, 平成19年7月20日 (2007)
39) IEEE Std 1471, "Recommended Practice for Architectural Description of Software-Intensive Systems -Description,"

IEEE (2000)
40) Bertalanffy, L. 長野 敬，太田邦昌訳.「一般システム理論」，みすず書房 (1973)
41) Shaw, M. and Garlan, D. "Software Architecture: Perspectives on an Emerging Discipline," Prentice Hall, 1996
42) DeMarco, T. and Lister, T. (Ed.). 児玉公信監訳.「ソフトウェアエンジニアリング論文集'80s」，翔泳社 (2006)
43) IEC 62264:2003. "Enterprise-control System Integration: Models and Terminology," IEC, ISO (2003)

## 付録

「期待」と「原要求」記述の小さな事例を示す。施主は、ある業務システムの運用管理者である。

**期待**

ビジネスのスピード、信頼性、継続性を保証すること。(1)ユーザが直接使うプログラムはサクサク動いて欲しい、(2)経営環境の変化に合わせてプログラムもその場で変更したい、(3)過去、現在のデータが失われないように保証したい。

**原要求**

(1)サクサク動くプログラム

取引件数、接続ユーザ数にかかわらず、一定の応答時間が保証されている。

(2)改修しやすいプログラム

どのプログラムのどこをどう改修するかが分かっていて、誰でも改修できる。

(3)データが失われない保証

大量データの確実で高速なバックアップコピーが確保できる。大量データの確実で高速なリストアが実施できる。

**パタンランゲージ（一部）**

### １　一定の応答時間

**背景**

接続ユーザ数が増えるに連れて、一定時刻にアクセスが集中する結果、応答時間が遅くなり、利用者の不満がある。

**問題**

接続ユーザ数が増えても、できるだけプログラムを変更せずに、応答時間を一定にしたい。

**解決策**

資源の競合を減らすこと。何が競合しているかによって、対策が異なる。

計算能力が競合している場合（略）、I/O能力が競合している場合（略）、データが競合している場合（略）。

**注意事項**

この解決策の適用前に、応答時間のプロファイリングを行い、応答時間遅延の原因を確認すること。

### ２　誰でも改修

**背景**

設計書の精度が悪い。設計者もすでに不在である。

**問題**

特定のベンダしか改修できず、時間と費用がかかる。

**解決策**

現行システムを作り直す。その際、合意された開発手法に従って、改修向けのドキュメントを作る。その内容は（略）

**注意事項**

現行データの移行方法を検討すること。データの移行パタンを参照のこと。