仕事の中でのコンピューター

応用情報処理II

2019/11/15

講師：新居雅行

本日の講義について

- 今日は、「講義」というより「セミナー」っぽく進めます
- 大学を卒業して働くようになった時に、多くの人はコンピューターとは切っても切れない状況になると思います
- 「コンピューターの使い方を知る」Iことで乗り切ることもありますが、本質を理解した上で高いレベルの活用が望まれています
- 今回は、そうした使い方というよりも、仕事をする上での要請をどのようにコンピューターはさばいて来たのかという話をします

3

コンピューターで仕事をする

仕事をするためのコンピューター

- コンピューターの利点「早い」「大量処理可能」「完全な記録ができる」点
  - いずれも、「人間に比べて」というところ
  - それだけに期待も大きく、SFでの独特の立ち位置（もちろん想像の世界だが）はもちろん、政策に絡むなど、社会の中で多方面に活用されている
- コンピューターの助けがあれば、仕事が効率良くなると考える
  - まさに、「表計算」をうまく利用すれば、時間のかかる計算もすぐ終わる
  - 電子的にファイリングされたデータは、一見すると「場所を取らない」
- さらに、コンピューターがあれば、人間ができないこともできるようになると考える
  - インターネットのECサイトで、「店員がいなくても」販売ができる
- 仕事に活用しない手はないぞ！

現在のコンピューターの立ち位置

- 当初は、ごく限られた人しか使えなかったコンピューター
  - それなりの性能はあったものの、実際にタッチできる人はごく一部
- パーソナルコンピューターの登場により「誰でも使える」というコンセプトが広がる
  - 表計算ソフトやワープロのような、手作業・紙作業の置き換えはわかりやすい
  - 従来の仕事をサポートすることが中心の時代
- インターネットの登場
  - コンピューターの利用範囲が爆発的に広がる
  - コンピューターによる新しいビジネス（経済）が広がることを期待
- スマホやタブレットの登場
  - 軽くどこにでも持っていくことができ、さらに誰でもが所有するようになった
  - 同時にAIブームも発生し、コンピューターが人間を置き換えることが真剣に議論されるようになった

問題点はたくさんある

- 現実には「使いづらい」と思っている人、そのように思ってしまう場面などはある
  - パッケージ化した製品やサービスは、自分好みに変更することに限界はある
  - 本質的に問題の大きなソフトウェアやシステムでも、結果的に使わざるを得ない
  - 問題がわかっていても、自分自身で解決ができない
- 想定している解決策は可能なのはわかるが、想定していないことはどうする？
  - 「自分でソフトウェアを作れ」と言われるが、そう簡単ではない
  - 自分で作って解決できるものならやるとしても、多大な時間をかけても思った結果にならないかもしれない
  - 「自分で作れ」は前世紀的な考え方。今時のITの世界はそんなに簡単ではない
- コスト感がわからない（お金がかかりそうなのはわかるが…）
  - 上記と同様な側面

問題点を解決できるとしたら

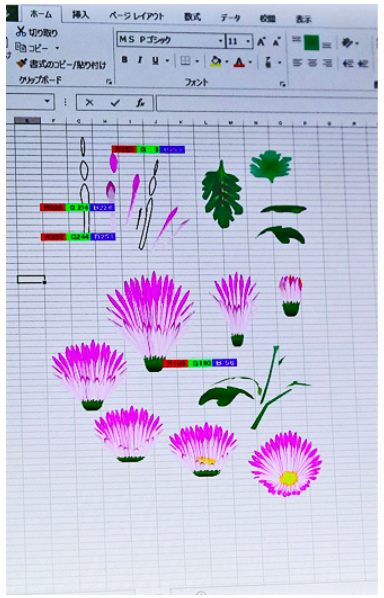
- 結果的に…
  - 情報革命とか言われても、自分ごとに思えない
  - プログラムを作ればいいと言われても、自分で到底作れるとは思えない
  - 勉強しようにも何をすればいいか分からない、勉強したことがすぐに陳腐化
- そうだ！システム開発は専門家にお願いすればいい！
  - 様々な専門職が生まれたが、ここでは「仕事にコンピューターを使えるようにする」人たちである「システムインテグレーター」（SIerと呼ばれる）を紹介したい
  - SIerにお願いすればいいという時代は短く、現在は存在まで否定される状況に（後述）
- だが…
  - ソフトウェアを作ると言う仕事がどんなものか想像できない
  - 作っている現場を見ても、ゲームやっているのとどこが違うのかよくわからない
  - 作られている途中のものが、本当に目的にかなったものかは一見すると判別できない

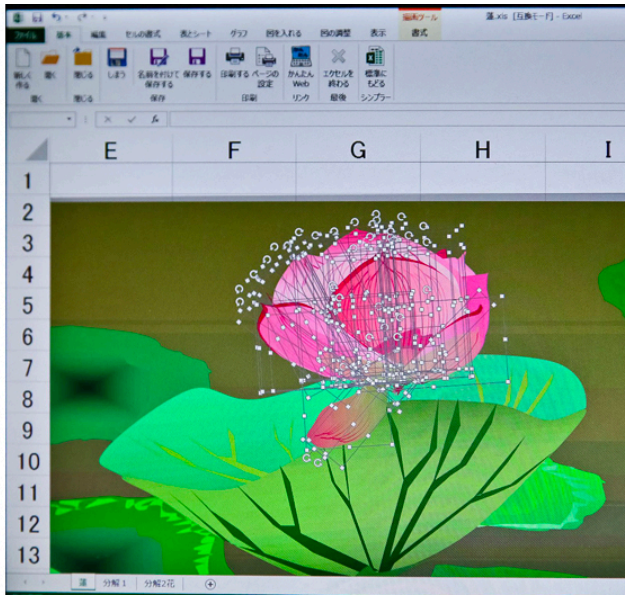
8

ソフトウェアを開発するとは

仕事にソフトウェアは必要か？

- 昔は「パソコンはソフトがなければただの箱」などと言われた
  - ハードウェア的な性能ばかり取り沙汰されるが、問題解決のためにはソフトウェアにポテンシャルがあるかどうかも大きな問題。ハード、ソフトの両面で評価が必要
  - Windowsも、Excelも、Chromeも、ソフトウェアである
- WindowsもExcelもあるのだから、それで十分では？
  - Excelで何でもできると豪語する人もいるが、まずは容易にできることとできないことは分離して考えるべき
  - 容易にできないことを実現する技術は素晴らしいのではあるが、それをチーム、会社、社会で共有できているかは疑問





Excelで絵画：素晴らしい技術だが、誰もができるわけではない
http://myvaio.sony.jp/sonyselect/special/140220/

必要になるソフトウェアとは？

- 業務を遂行するための専用のソフトウェアが必要だと考える
  - Excelで行っている1時間の作業が、専用ソフトウェアを作ると、30分で終わるとしよう
  - 従業員が1万人いたら、5000時間の節約になる。時間単価が3000円としても、1500万円の節約になる。この作業が1年に1回でも、1000万円かけてソフトウェアを構築できるとしたら、毎年500万円の収益になる（これはごく簡単なモデル）
- しかしながら、ソフトウェアの構築や運用（稼働させること）は自社ではできない
  - そうだ！SIerに頼めばいいではないか！
  - SIerは基本的には「プロ集団」、だから、ちゃんとやってくれるに違いない
- そういえば、自社にシステム部門があったぞ！
  - なぜか、日本のシステム部門は、自分でソフトウェアを作らない（海外は作っている）
  - 結果的に、システム部門からSIerに丸投げする

ソフトウェアを作るには

- 狭義には「プログラミング」をすること
  - あるいは、そのように捉えている人があまりに多い
  - その結果、プログラミングをマスターすれば、ソフトウェアが作れると思っている人が多数発生する
- プログラミングは、あくまで「製造プロセスの1つ」である
  - 製造しなければ意味はないのはもちろんだが、品質良く製造するためには、単にプログラミングに人を割けば良いということではないことが、長年の研究でわかっている
  - 大まかにいえば、次の点が重要であり、プログラミングは、設計結果を元にした実装作業であり、開発プロセスの1つに過ぎない
    - どのように作れば良いかという「設計」の問題
    - 作ったものが正しいかどうかの「検証」の問題
    - 使い続けることを実現するための「運用」の問題

それじゃあ、天才プログラマーって？

- 確かにいる。Linux/Gitの開発者であるリーナス・トーバルスは代表例
  - コンピューターに長けた人を「天才プログラマー」と簡単に呼びがちであるが、本質的な褒め言葉ではない
  - 一部の天才は、「設計」「検証」「運用」を全部頭の中で解決して、そしてプログラミングを行い、品質の高いソフトウェアを作り出す。これは確かに素晴らしいが、一般には「設計」が先進的、かつ確実な検証を行っていることが普通
- 天才でないと作れないものは「工業製品」とは言えない（あえていえば作品）
  - 工業製品には、一定の計画された資源投入で、計画に従ったアウトプットが望まれる
  - 例えば、いつ仕上がるか分からない業務用ソフトウェアを、会社の経営計画に組み込めるのかという問題
  - そこで、一定の教育レベルのエンジニアでも、品質の高いソフトウェアを一定の納期で開発する技術が開拓されて来た（その分野は「ソフトウェア工学」と呼ばれる）

https://ja.enterpriseunix.org/20-facts-about-linus-torvalds

13

ソフトウェア工学の視点

ソフトウェア工学とは

- ソフトウェアを生産物として捉え、開発の初期段階からユーザーがソフトウェアを利用できるようになるまでの各側面で様々な手法を駆使し、品質を確保し、納期を確定し、コストの産出を可能にするための知識体系。（参考：浅井「実践的ソフトウェア工学」）
- ソフトウェア工学が必要とされる背景
  - 大規模化、複雑化するソフトウェア開発
  - 社会インフラを担うようになったIT業界
  - セキュリティ上の脅威に否応無しに曝される
  - ビジネスが成立するためのコスト構造、法律等に逸脱しない労働環境

ソフトウェア工学の要素

- ソフトウェアの品質
  - ソフトウェア工学の最も上位、かつ中心的な目標
    - バグがない（少ない）
    - 必要十分な性能
    - 動作可能な条件が合理的
    - 改変しやすい、など様々な要因がある
- 納期とコスト
  - 言い換えれば、「計画がきちんとできている」
  - その本質は、問題点が解決されている、あるいは問題点を克服できる目処が立っている
  - 加えて、「計画がきちんと遂行できる」
    - いい加減な納期設定やコスト計算は、プロジェクトの失敗を招き、開発プロジェクトだけでなく、顧客にも多大な損失を発生させる可能性がある

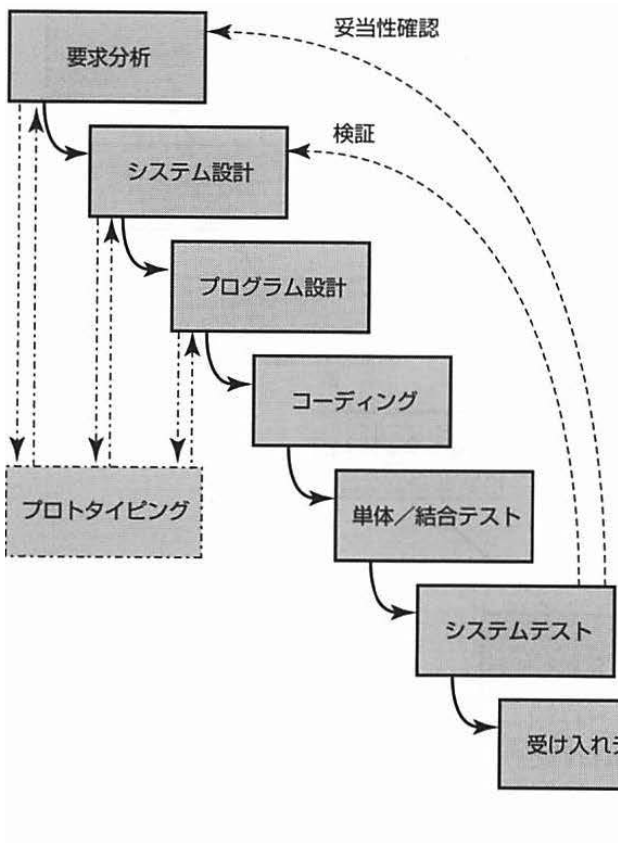
解決策を導く基本的な手法

- 抽象化
  - 問題の本質を捉えるための基本的な手法
  - 様々な解決手法とのマッチング
  - 抽象化した結果を記述したものが「モデル」
  - 「階層化」はコンピューターの動作によく適合する
- 分析と表現方法
  - ダイアグラムで表現するUML等の手法があり、その結果をもとに分析する
  - 形式手法やモデル検査といった厳密にモデルを定義分析する方法も
- アーキテクチャー
  - 部分や全体が健全に成り立つための構造設計（参考：プリーガー「ソフトウェア工学 理論と実践」）
- デザイン
  - 人間とシステムとのインタフェースを多面的に捉える
  - 古い時代は「プロトタイピング」と呼ばれた

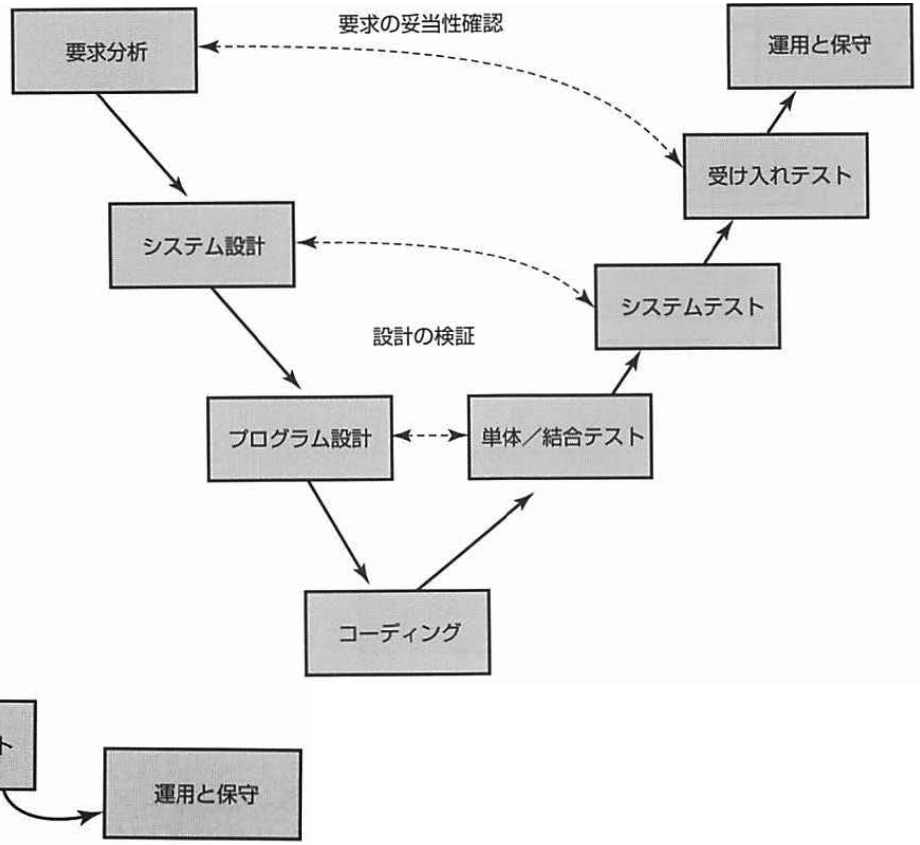
開発プロセス

- 開発の進行を分割して表現したもの
  - 過去の様々な開発経験をもとに、作業の段取りが抽象化されたもの
  - 常に「その通り」にやっているわけではないが、概ね「その通り」にされている
- 様々な開発プロセス
  - ウォーターフォール
  - プロセスを順番に実行する手法
  - 前半を「上流工程」と呼ばれるもとになった考え方
  - Vモデル：ウォーターフォールの工程間の関連付け
  - スパイラル：行程を繰り返すことで完成に近づける
  - アジャイル：短期間で一定のゴールを達成し、検証して開発を進める

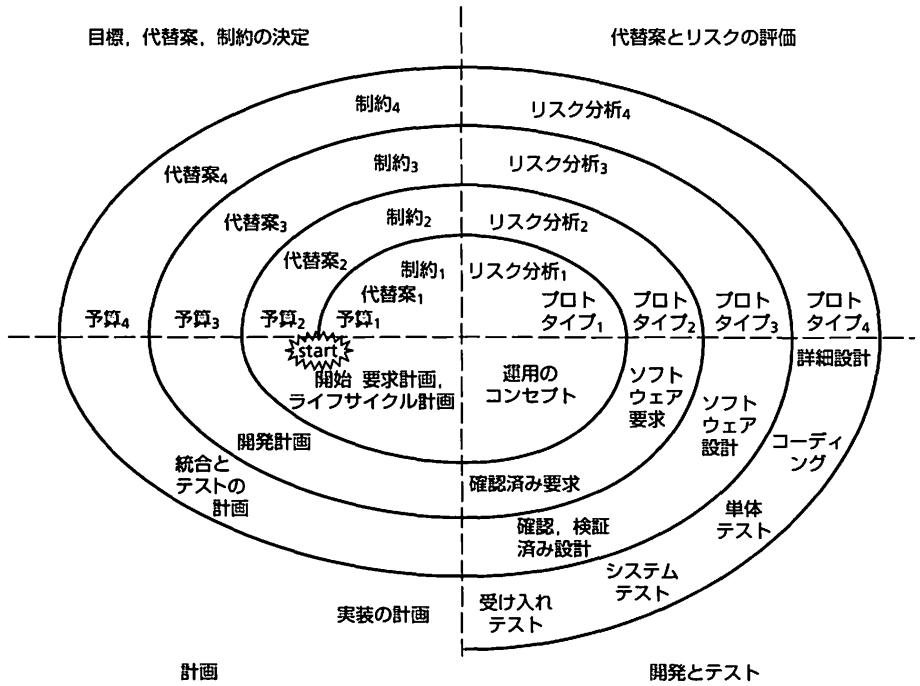
ウォーターフォールモデル



V字モデル



スパイラルモデル



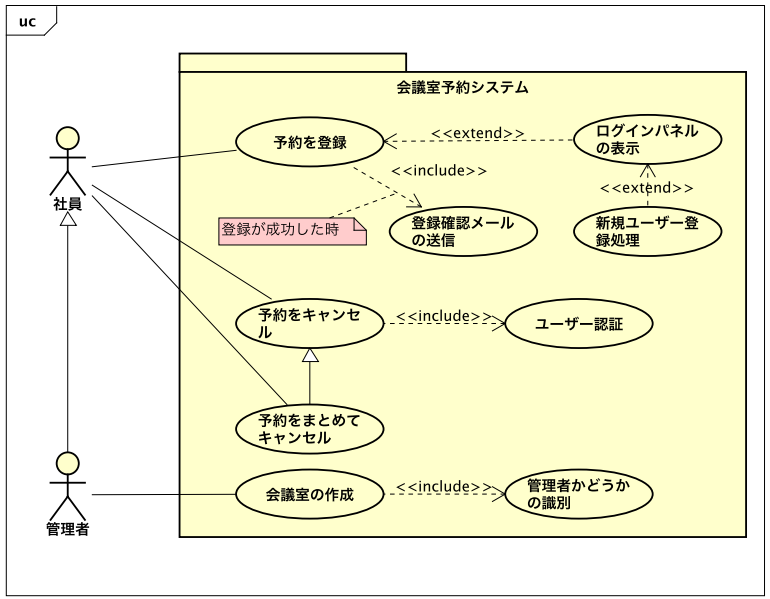
参考：プリーガー「ソフトウェア工学 理論と実践」

「コードを書く」のはどの範囲？

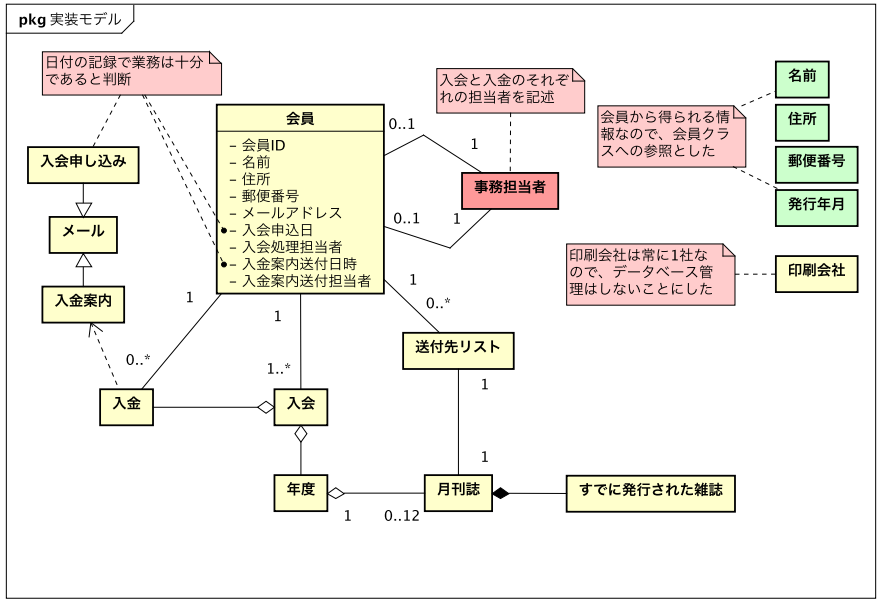
- 直接的には「コーディングプロセス」
  - つまり、設計をした結果をもとにコーディングをする
  - 間接的には、ほぼ全て
  - 現在は、テストのためのコードを記述して、システムのテストを自動的に進めるなどもある
- なぜかコンセンサスが揺らぐ基本用語
  - プログラム：抽象的な意味で命令を集めて一定の目的を達成できるようにしたもの
  - コード：プログラムを具体的に記述したもの
  - コーディング：コードを作成すること
  - プログラミング：意味はプログラムを作成することだが、「手続き言語のコーディング」でないと違和感を持つ人が多い
- 設計を含めた「より高度な作業」であると思う人が一般的
  - 現実には、設計と独立したのがコーディング作業

コーディングまでの作業（上流工程）

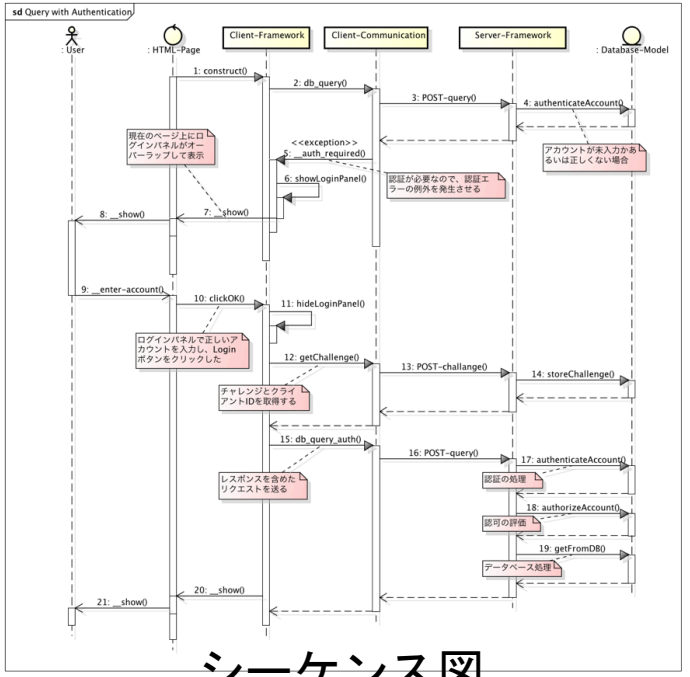
- 要求獲得、要求記述、要件記述
  - システムで何をしたいのかから始まり、それを実現するために何を作ればいいかを確定させる
- 設計
  - 一般にはダイアグラムを使ってビジュアルに表現することで、視認性を高める
  - ルール（UMLが代表的）に従って記述することで、曖昧さを排除する



ユースケース図



クラス図



シーケンス図

コードを作った後の作業

- デバッグ
  - 不具合（バグ）を取り除くこと。通常は、コーディング初期から行われる
- テスト
  - 仕様に合致するかを確認する作業。手動でやることも多いが、最近は自動化が盛ん
- 配備
  - 作成したソフトウェアを用意したハードウェア上に展開し、システムとして稼働する状態にする
- 運用
  - 作成したシステムが日々滞りなく稼働させるようにする作業
- メンテナンス開発
  - 過去に作成したシステムに対して、仕様の変更や追加が発生し、既存のシステムをベースに開発を進める
  - 現在の開発現場では、多くがこのような側面を持ちながら、エンジニアリングよりも勘と経験でなんとかこなしていると言う負の側面もある

22

ソフトウェア開発を業務とする会社

ソフトウェア開発を請け負う

- 20世紀後半より存在する業務形態
  - 多数のエンジニアを擁する。一方、営業も重要
- 請負契約上、どうしても、作る前に値段を決めないといけない
  - そのためにも、設計は重要だが、設計にも時間がかかる
  - 結果的に、勘と経験値で決めるとか、そういう雑な話になりがち
  - 見積もりをする手法も開発されているが、今現在でも正確に見積もる方法は模索されている
  - 一方、値段が決まった瞬間、人数×開発期間が固定されてしまう
- そして、契約する
  - 昔は、契約そのものがないとか、曖昧とか、そういう問題があった
  - 現在は、以下のどちらかの形態で契約する
    - 望む結果を顧客が得られることを保証する「請負契約」
    - 望む結果を顧客が得られるにはこれくらい働けばいいかなという労働提供を保証する「準委託契約書」

請負業の存在と矛盾

- 人数を投入しても開発効率は上がらない
  - 「人月の神話」などと呼ばれる。既に、1975に言われている
- 損はしたくない、損はできない（会社潰れます）
  - 見積もり費用は当然ながら、高めに設定し、何かあっても「損しない」ようにする
  - 結果的に「高価な割にちゃんと使えない」と顧客は思いがち
- 計画はするが、その通りに行かず、普通はより時間がかかる
- プロマネ（プロジェクトマネージャー）がしっかりしていればできるはず
  - マネジメント論になるが、そう簡単には行かない。「責任者出てこーい！」ではかわいそう
  - とは言え、技術力のあまりないエンジニアがプロマネをやる事例も多く、なんだかsick!
  - 現場では、リーダーシップよりもメンバーシップが重視されるが、やはり責任者は荷は重い
- SIerは最新技術を駆使する会社と思いきや、旧態然なところも多い
  - 結果的に人材が定着しない。できるやつは他に行ってしまう。予算がない。冒険しない

なぜ計画通りに行かないのか

- 顧客は、自分が作りたいものを明確に説明できない
  - 矛盾があると思うかもしれないが、これが事実
  - 顧客は、イメージで話をし、具体的なことを尋ねるとなぜか言葉が止まる
- 作りたいもの＝「要求」
  - 要求を明確化し、明文化し、システムを構築する人が様々な場面でそれを参照し、顧客が欲する仕組みをシステムに実装しなければならない
  - しかしながら、要求が不明確、曖昧、あるいは、記述さえされていない
- さらに、顧客は、作ったものが、欲しかったものかどうかを検証することができない
  - これも、要求が曖昧だから
- 顧客は「後出しジャンケン」をする
  - 開発されたものを見て、「こんなんじゃない」「こんな機能も入れて」「そこはそうではなくて…」
  - 昔の曖昧な契約書ではSIerが泣かされていたが、現在の契約では単に喧嘩別れもできる

開発がうまく進まなくなると

- 開発が遅れたりすると、人を投入したり、エースを投入するが…
  - 通常は、パフォーマンスはさほど上がらず、混乱するだけ
- 計画遅れ、顧客からの要求変更が重なると…
  - チームメンバーがネガティブになり、さらにパフォーマンスが落ちる
  - うつ病で寝込む奴が出てくる（コンピューター業界は他業種より多いとされている）
  - 本当に蒸発する奴も出てくる（なぜか「ハワイに逃げた」という噂が飛び交う）
- そして、本当に何も進まなくなると「デスマーチが鳴り出す」
  - デスマーチという曲があるわけではないが、そういう音楽があることを業界全員知っている
  - もうどうしようもない状態になることを「デスマーチが鳴る」と呼ぶ
- さらに、本質的な問題として、「多重下請け構造」も、デスマーチへ到る一因である
- ちなみに、その後どうなるか？
  - 会社が夜逃げ、あるいは、開発会社が「ごめんなさい」するなど

27

これからのコンピューター利用

SIERは不要なのか？

- Slerの仕事の進め方自体がまずいのではないかとも言える
  - 契約の厳格化、要求定義に時間をかける、開発効率を高めて仕様変更をある程度は受け入れられる体制にするなど、多くの会社で様々な努力
- そもそも、顧客に原因があるのではないかとも言える
  - 要求を明確にしていないのを、開発側の責任にはできない
  - 明確にできないことを、前提としなければならないのではないか
    - そうした状況でも開発が進むように、短い期間ごとに検証を行うようなアジャイル開発も一般化
- つまり、Slerの業務内容が大きく変わって来ている
  - 2010年以降、その傾向が強くなって来ている

これからの開発とは？

- 従来的な特定業務用のシステム開発は残る
- 標準化された業務は、既に作られたシステムを使う
  - クラウドやSaaSが発展し、現実的になっている
  - 大きな会社ほど、様々な法律に縛られた活動が必要であり、その点では、業務に関して会社ごとの違いはそれほどないという見方もできる
- 紙ベースの作業の置き換えから、DX(Digital TRANSformation)へ
  - 紙をパソコンに置き換えても、単にペンをマウスとキーボードに持ち替えただけであり、業務内容の本質的な違いはなく、効率の追求でしかない
    - その結果、本質的でないIT運用が目立つ（14回目の講義で紹介する「ネ申エクセル」など）
  - コンピューターやインターネットを巻き込むデジタルワールドの特性を活かせば、大きな改革ができるのではないかというコンセプトがDX
    - スマホで簡単に商品が購入できるなど
    - 従来の会社の方向性を大きく変える必要はあり、着地点は遠くに設定されたと言える

業務システムの標準化

- 既に作られたシステム
  - ERP(Enterprise Resources Planning:統合基幹業務システム)などと呼ばれ、SAPやOracleなどが製品を持つ
  - 大企業は、一般にはEPR製品を導入して、独自開発よりも低コスト、かつ多様な機能を利用できるようにしている
    - ここで独自なことをしても、競争力の原動力にはならないことを理解
- SAPがリードしているのは、「ドイツ自身が標準化された社会である」*1
  - 結果的に、標準化したシステムの需要が盛り上がる
  - 他国や、あるいはシステム化の弱いジャンルでも、大手ベンダーをコアにして、標準化が進み、それを企業が導入することで、社会の標準化が進む
- もはや、開発をすることからスタートしている場合ではない
  - 既にあるものに業務を適合させ、状況によっては部分的なカスタマイズをするのがトレンド
  - Quick Startが当たり前の時代、スクラッチから作るのは限られたシステムになる

*1 SAPの成功： ドイツの制度環境からの一考察 同志社大学客員教授、フランス国立労働経済社会研究所（LEST）、ドイツ日本研究所（DIJ）客員研究員　山内 麻理 https://www.ipa.go.jp/files/000068596.pdf

では開発は無くなってしまう？

- そんなことはない
  - 少なくとも、何らかのソフトウェアは必要なので、開発そのものは未来永劫続けられる
- 従来のSIerのビジネスモデルが否定された？
  - 「否定された」と言う意見もあるが、需要は全くなくなるわけではない
  - 標準化されていない作業はどの会社にもある
    - むしろ、他社と違うことをやっていることで競争力が発生するなら、そのためのシステムは自分で用意しないといけないことにもつながる
  - 仕事の内容が変貌しつつある
    - 比較的最近までは、「なんでも１から作ります」的なことがよくあった（その方が儲かる）
    - しかしながら、既存の仕組みをうまく利用しつつ、世の中にないものは作るなど、文字通りのインテグレーションを真面目にやるようなビジネスが中心になりつつある
  - 実は、DXを真面目にやらないといけないのは、コンピューター業界だと言われている
    - 紺屋の白袴的な状況は、もはや笑い話にしかならない

まとめ

- コンピューターを利用することで、仕事が効率的に進み、できなかったことができるようになると言うコンセプトは正しい
- しかしながら、その背景では、ソフトウェアを開発すると言う作業が発生する。自分で開発できればそれに越したことはないが、通常は専門業者に依頼する
- ソフトウェア工学という世界では、品質を高め、計画通りに作業ができるような仕組み研究している
- ソフトウェアの開発は特殊技能であり、そのために請負開発が一般に行われているが、開発プロジェクトの多くは「失敗した」と言われており、大金が動く割には痛い業界でもある
- 最近は、標準化された業務をこなす、標準化されたソフトウェアを導入する動きが強くなり、必ずしもスクラッチから開発するわけではない