- 組込みソフトウェアの開発に、なぜモデルを使うことが必要とされているのかを学ぶことから始めましょう
- 「なぜモデルが必要なのか」について、いっしょに考えてみましょう

1. なぜ、モデルが必要なのか？

1-1. 組込みソフトウェア開発の特徴

開発のミッション

- 自然現象や物理現象を相手に
  「人間が容易にできないことを装置を使って実現する」
  - 携帯電話、デジタル家電、自動車、複合機、産業用装置etc.

必要な技術

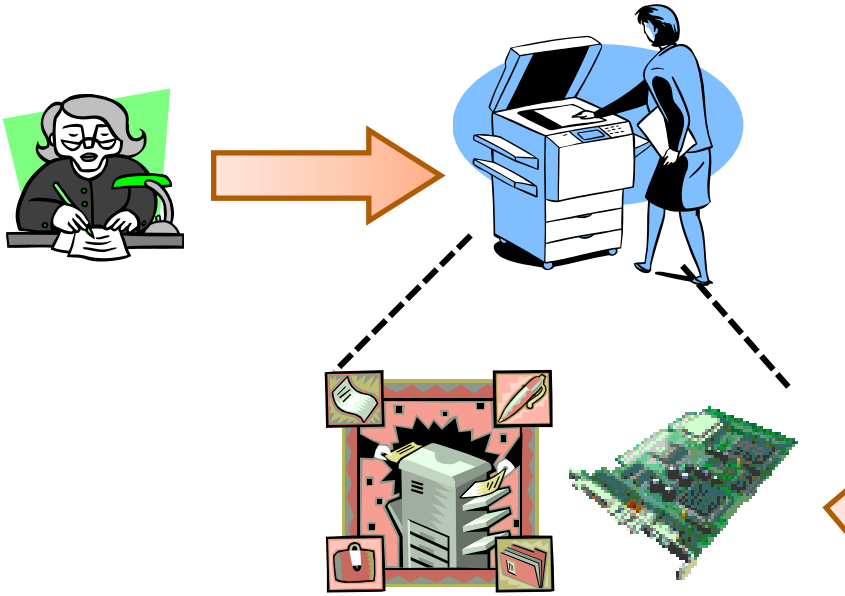
- 机上だけでは予測できない問題が多い
  - 環境制約、物理的制約、ハードウェア制約、コスト制約など
- 現場で試行錯誤する中で構築される「要素技術」が重要
  - 制御戦略、アルゴリズム、ハードウェアの使い方、etc.

必然的に…

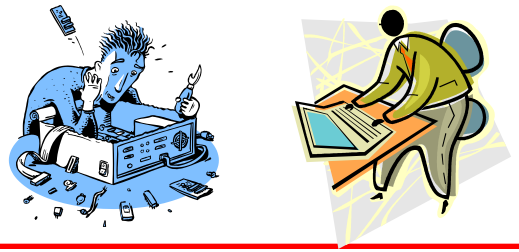
- 要素技術主導の、現物合わせで作り込む開発スタイルが主流
  - プログラミングして動かしながら試行錯誤的に開発
  - 最終的に動いたものが仕様となり、量産のソフトウェアとして出荷

1-2.　組込みシステム開発の例：複合機

- 開発のミッション（狙い/目的）
  - 短時間で、大量の書類の複製を可能に！（人手では到底不可能）



「現物合わせ」で作り込む開発スタイルが常態化



必要な技術　温度/湿度の制御方式、用紙の搬送、画像の転写方式、少ないROM/RAM上での開発テクニック、短い期間で開発する独自の方法など、技術主導で開発を実施

結果として、
組込みソフトウェア開発は、
動かしながら試行錯誤的に開発され、最終的に動いたものが仕様になりやすい

1-3. 組込みソフトウェアの陥りがちな品質

- 現物合わせや試行錯誤は品質に影響します
  - 煩雑＆複雑なロジックを生みます
    - 膨大な試行錯誤の結果が、そのまま定着してしまう
    - さまざまな知識の集大成なので、理解するのが大変
  - 意味が分からないロジックになりがちです
    - どんな目的でそのロジックになっているのかがわからない
  - どこに何が書いてあるのか把握できない
    - プログラミング言語で記述されたソースコードが主体
    - ソフトウェア全体を説明するものが存在しない

1-4. 組込みソフトウェアが抱える問題

- 品質が悪いと次のような問題が発生します
  - 修正や追加に時間がかかる
    - 詳細なソースコードしかなく、しかもロジックが複雑で難解
    - 内容を理解することに、多くの時間が必要
  - 障害（バグ）の発生が収束しない
    - 障害：プログラムに含まれる誤りや不具合
    - 内容を推測しながら試行錯誤を重ねて作るため、想定しないところでバグが多発する
- 問題の対処をふたたび現場合わせで…
  - 負の連鎖が繰り返される
    - 場当たり的なソフトウェアが作られる
    - バグ対応に追われると時間がなくなる
    - さらにロジックが複雑・難解になる
- 品質はさらに悪化し、生産性も低下し続けます

1-5. 組込みソフトウェアのあるべき姿

- 組込みソフトウェア開発のもう一つの特徴
  - 長期間（数年～数10年）使い続けられる
    - めったに見たり変えたりできない機器の中身を扱う面白さがある
  - 搭載される製品や装置の進化と共に新しくなり続ける
    - 新しい技術を世の中に出る前に使うチャンスに恵まれている

- 「分かりやすいソフトウェア開発」が重要です
  - 他の人も直しやすい
    - 後の追加修正を想定した直しやすさ
  - 変化に対応する余地を生み出す
    - 長期間の使用に耐える頑健さ
    - 性能向上のために要素技術の試行錯誤も想定する

1-6. 分かりやすいソフトウェア開発とは？

- 大局的なソフトウェア記述の導入
  - ソースコードは、コンピュータに対する詳細な命令の集合
    - 目的・意図の理解、プログラム全体の俯瞰・流れの把握などには向かない
  - 別途、意図や全体像が把握できるものが必要
    - ソースコードの短所を補う別の表現をした記述資料を作る

- 手が入れやすく影響範囲が分かるソフトウェアへの移行
  - 動かすことだけを考えた、手続き主体の巨大なロジック
    - 修正や追加の際に、どこに手をいれたらよいか分かりにくい
    - 修正の影響範囲がわからない、影響範囲が広範に及ぶ
  - 長期的なメンテナンスや機能追加をしやすくする
    - 手がいれやすく影響範囲が分かる設計手法を使う
    - 設計に対応したソフトウェアを作る実装方法を使う

1-7. 部品化と抽象化

- 大きくて複雑な問題をうまく扱いたい
  - 大きな問題を小さい問題に分けたい
  - 複雑な問題を簡単な問題に分けたい

- 部品化
  - 大きな仕事を分けて、他のモノや人に手分けすること
  - 手分けをした仕事を組合わせて、大きな仕事をこなすこと
  - 分けたものやその組合わせを分かりやすい表現を使って表す

- 抽象化
  - 複雑な問題から興味がある問題を抽出して、取り扱いやすくする
  - 興味のあるところを抽出する（抽象）
  - 興味がない部分を削る（捨象）
  - 抽出されたものが分かりやすいような表現を使って表す

塑造で作る塑像

彫刻で作る彫像

1-8. モデルとモデリング

モデル

- 部品化や抽象化によって得た情報をつかった表現
  - 個々の部品や抽象を分かりやすく表現したもの
  - 部品や抽象の組合わせを分かりやすく表現したもの
- 分かりやすい表現、組合わせやすい表現
  - 図、式、絵、模型、ブロック、など

モデリング

- モデルを作る行為
- 部品やその組合わせを表現する
- 抽象やその組合わせを表現する

1-9. モデルを使って開発する

- ソフトウェアのモデル
  - ソフトウェアの設計図
    - ソフトウェアの複雑な構造や振舞いを、理解しやすくする
    - 図や数式などを用いて興味の対象に絞り込んで、よく分かるようにする
  - 大局的な記述
    - 実装の詳細には依存しない、ソフトウェアの動作の目的や意図を示す
    - 細い構造を大きなまとまりに分割したり、細かい動きを大きな動きにまとめたりする

- ソフトウェア開発のモデルで扱いたいこと
  - 開発する目的や実現したいこと（要求や機能）
  - 構成する要素とそのつながり（構造）
  - 処理の流れやできごとと処理の関わり（振舞い）

1-10. 部品を使って開発する

- ソフトウェア開発に使う部品
  - 特定の目的を実現する小さな範囲のソフトウェア
  - 他から利用できるよう独立性・再利用性を高めたもの

- 部品を使った開発
  - 複雑で大きなソフトウェアを、部品を組合わせて作る
  - 部品が詳細を隠すので、大きなソフトウェアが設計しやすい
  - 部品の独立性が高ければ、変更や追加がやりやすくなる

- 部品の組合わせを使ってモデルを作ると効果的です
  - 部品間のつながりには2次元的な広がりがある
    - プログラミング言語では俯瞰する表現は難しい
  - 図を使うと部品のつながりが把握しやすい
    - つながりの有無も図だと直感的に表現できる

1-11. まとめ

- 今の組込みソフトウェア開発には問題が山積しています
  - プログラミングが先行し、試行錯誤的開発が繰り返されている
  - 作られたソフトウェアは分かりづらい、直しにくい
  - そのまま直し続けると、どんどん品質は悪化し、生産性も低下する

- この問題を解決するためには...
  - 試行錯誤で作り込むソフトウェア開発だけでは立ち行かない
  - 追加や修正を想定した「分かりやすいソフトウェア開発」が必要

- これからは「モデル」と「部品」を使った開発が有効
  - モデルを使って、ソフトウェアを大局的に把握し、整理する
  - 部品を組合わせた開発で、規模や複雑さに対抗する

理解度チェック（１）

- なぜ、モデルが必要なのか？

1. なぜ、近年の組込みソフトウェア開発では、品質が悪化しがちなのでしょうか？
   a. 理由は？
   b. どのように悪化しがちですか？
2. 分かりやすいソフトウェア開発を行うための大きな２つのポイントとは何でしょうか？
3. 「モデル」と「部品」の関係はどういうものでしょうか？