不具合に関する知識の抽出に関する研究

JaSST'08 Tokyo
2008年1月30日(水)

電気通信大学大学院
河野 哲也 / 西 康晴

プレゼンテーションの流れ

- 研究の背景
  - 現状の問題点
  - 提案内容
  - 適用例
  - まとめ

ソフトウェアによるトラブルが後を絶たない

- ソフトウェアの不具合は深刻な問題を引き起こしている
  - 特に組込みソフトウェアが顕著
    - 新幹線のATCのプログラムミスで機能停止
    - 携帯電話の電話帳情報が消失
    - 自動車のエンジン制御プログラムに不具合
- 過去に発生した不具合には同じようなものが多い
  - 新規のメカニズムによる不具合よりも
    過去の不具合と同じようなメカニズムによる不具合が圧倒的に多い
    - この不具合は前にも見たことがあるなぁ
    - この不具合は以前作りこんだことあるなぁ
      - 同じような不具合ばっかりだよね。。。

不具合に関する知識を活用する

- 不具合を分析し、メカニズムが明らかになれば市場での不具合を減らすことができる
  - 不具合に関する本質的な知識を抽出する
    - 過去の不具合を分析しメカニズムを明らかにすることで知識を抽出する
  - 不具合に関する知識を活用する
    - 開発で
      不具合に関する知識を活用することで
      不具合の作りこみの未然防止が行える
    - レビューやテストで
      不具合に関する知識を活用することで
      多くの不具合を検出できる

現状の問題

- 不具合に関する知識を十分に活用できていない
  - 効果的に不具合の未然防止が行えていない
  - 効果的に不具合の検出が行えていない
    - そもそも抽出した不具合知識に問題がある
- 不具合に関する本質的な知識を抽出できていない
  - 不具合を作り込むメカニズムが明らかになっていない
    - メカニズムを反映した知識を抽出できていない

研究の流れ

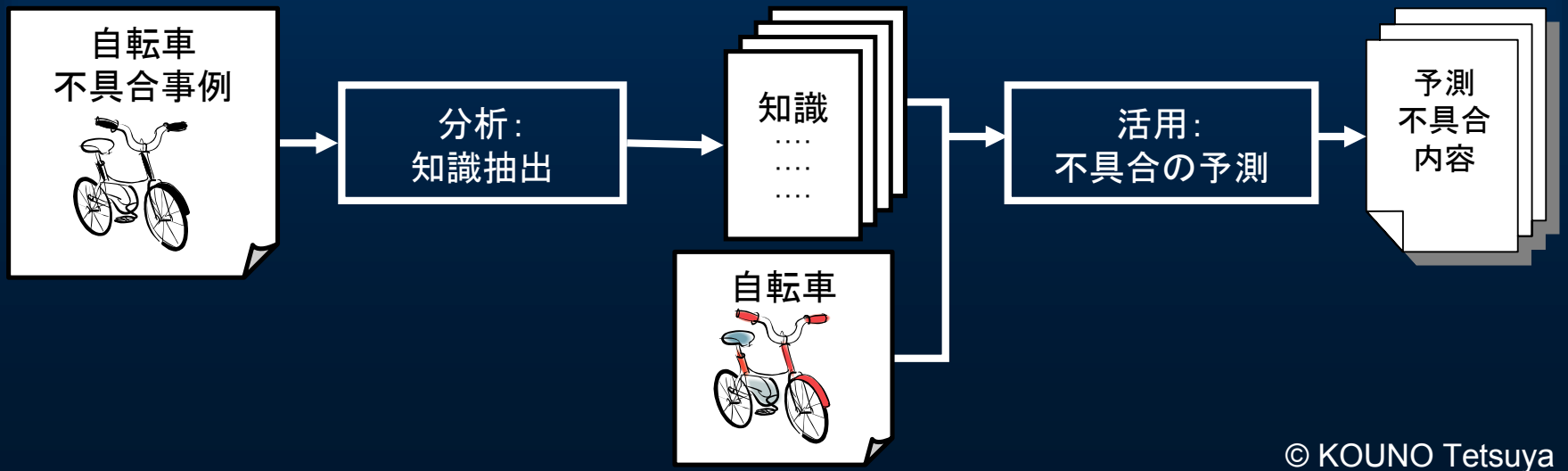
- ソフトウェアにおける知識抽出の問題点を述べる
  - ハードウェアの知識活用の例を説明する
  - ソフトウェアにおける知識抽出の問題点を明らかにする
- 不具合に関する本質的な知識が抽出できるようにする
  - 不具合を作り込むメカニズムを明らかにする
    - ロジカル･アフォーダンスの提案
  - 不具合を分析するための着目点を提案する

プレゼンテーションの流れ

- 研究の背景
- 現状の問題点
- 提案内容
- 適用例
- まとめ

ハードウェアの知見を学ぶ

- ハードウェアにおける不具合知識の活用の考え方を解説する
  - ハードウェアにおける不具合に関する知識の抽出と活用の流れを把握する
- ソフトウェアにおける問題点を明らかにする



自転車のチェーンの不具合

- 自転車のチェーンが上手く機能しない
  - きしむ音がする、スムーズに回転しない、・・・など

きしむ!!

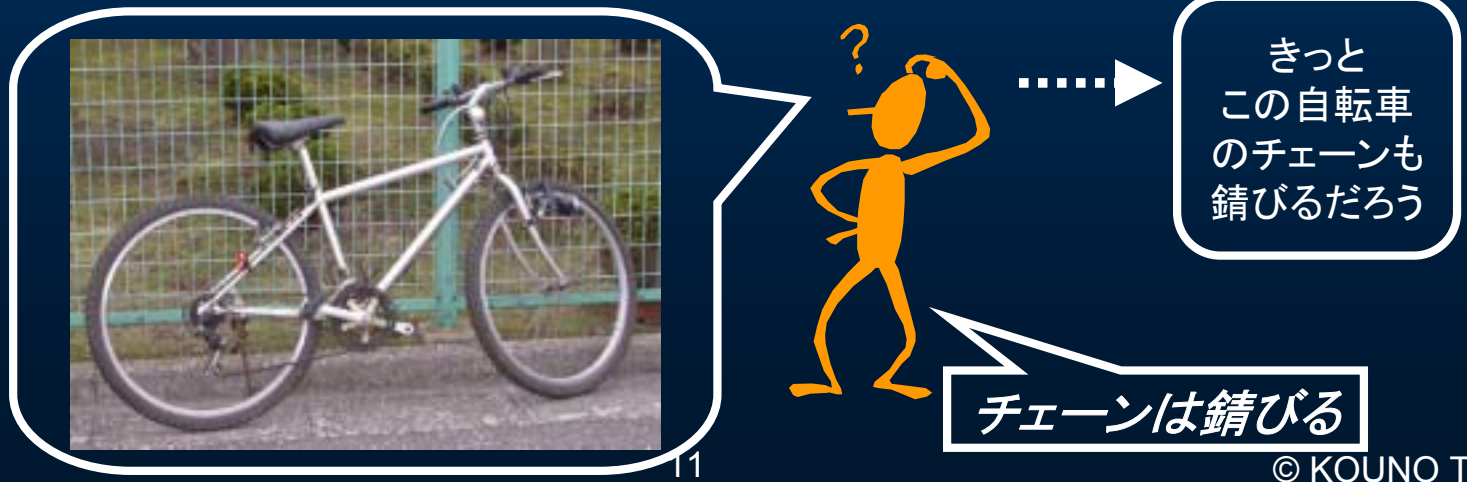
スムーズに動かない!!

不具合の分析、不具合知識の抽出

- チェーンの不具合を分析してみよう
  - どうもチェーンが錆びているようだ!!
- 不具合知識の抽出
  - チェーンが錆びているので、
    不具合知識“チェーンは錆びる”を抽出する

不具合知識の活用

- 不具合知識を活用する
  - 不具合知識“チェーンは錆びる”より
    違う自転車の不具合を予測してみよう
- この自転車にもチェーンはついているようだ
  - 不具合知識“チェーンは錆びる”より
    この自転車のチェーンも錆びることが予測できる



不具合知識の再利用の検討

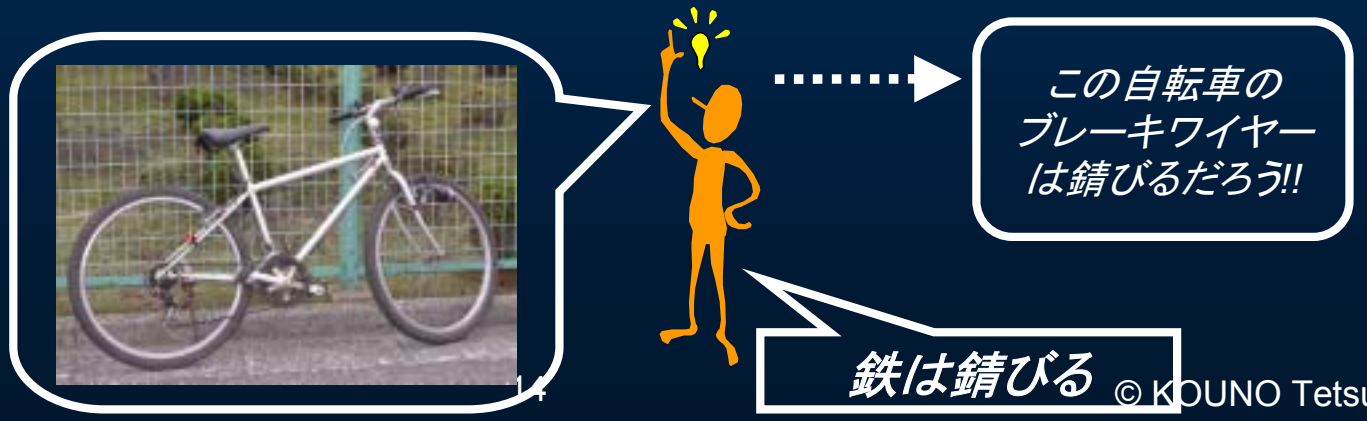
- 不具合知識の活用を検討する
  - この自転車のチェーンはチタン(錆びない材質)で
    できているようである
  - この自転車のチェーンは錆びない⇒知識活用の失敗
- “チェーンは錆びる”という不具合知識が適切ではない
  - 的確に不具合を予測するためには、
    不具合に関する本質的な知識を抽出しなければならない

不具合の分析、不具合知識の抽出

- チェーンの不具合を分析してみよう
  - チェーンだから錆びるわけではないようだ
  - よく考えてみると、チェーンの材質は鉄のようだ!!
- 不具合知識の抽出
  - チェーンの材質が鉄、鉄は錆びる
  - 不具合知識“鉄は錆びる”が抽出できる

不具合知識の活用

- 不具合知識を活用する
  - 不具合知識“鉄は錆びる”より
    先ほどの自転車の不具合を予測してみよう
- 鉄の材質を使っているのは、どの部品だろう？
  - ブレーキワイヤーが鉄でできているようだ!!
  - 不具合知識“鉄は錆びる”より、
    この自転車のブレーキワイヤーが錆びることを予測できる
    - 単純に考えただけでは、チェーンとブレーキワイヤーはつながらない



不具合知識の活用

- 不具合知識“鉄は錆びる”より
  自転車以外の不具合を予測してみよう
- 例えば、バイクのホールや三輪車のフレームが鉄でできていれば
  - もちろん錆びることが予測できる
- 例えば、ブランコが鉄でできていれば
  - もちろんこちらも錆びることが予測できる
    - 単純に考えただけでは、自転車とブランコはつながらない




きっと
これらは
錆びるだろう!!

鉄は錆びる

物理・化学メカニズムに着目する

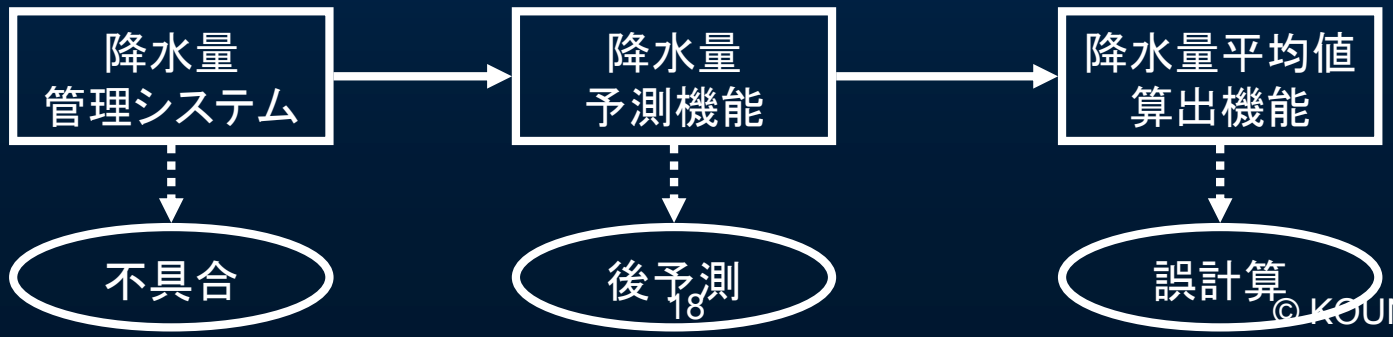
- 的確に不具合を予測するには
  不具合知識“鉄は錆びる”が良いだろう!!
  - ハードウェアの不具合は物理・化学法則に従って発生する
    - 例：鉄は錆びる、L字構造は疲労破壊が起こる、など
- 本質的な不具合知識とは物理・化学メカニズムである
  - 分析の対象とすべき要素単位を「材質」や「構造」に設定する
    - 例：鉄、L字構造
      - チェーンやブレーキワイヤーなどの“部品”に設定すると上手くいかない
  - 要素単位が望ましくない状態を一般化・抽象化して表現する
    - 例：錆び、疲労破壊

本質的な不具合知識を抽出できていない

- ソフトウェアにおける現状の問題
  - 本質的な不具合知識を抽出できていない
- 現状の不具合知識の抽出における問題点
  - 分析の対象とする要素単位を「機能」に設定している
    - 例えば、
      “出力機能が誤動作”や
      “入力機能が動作しない”
      - 「チェーンは錆びる」のような
        不具合知識を抽出していないだろうか

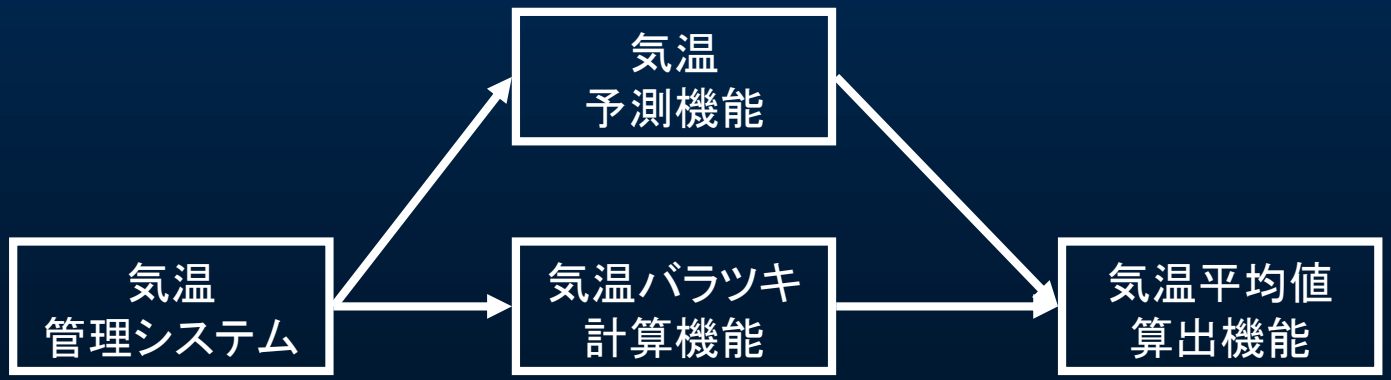
具体例：過去の開発の不具合事例を分析

- 過去の開発の不具合事例：
  - 降水量管理システムの開発で不具合があった
  - 降水量予測機能の予測が正しくできなかった
    - 降水量を予測するには前年の降水量の平均値を使用している
  - 前年の平均値算出機能が正しく計算できなかった
- 不具合知識：
  - 管理システムに不具合があった
  - 予測機能が誤予測を起こす
  - 平均値算出機能が誤計算を起こす



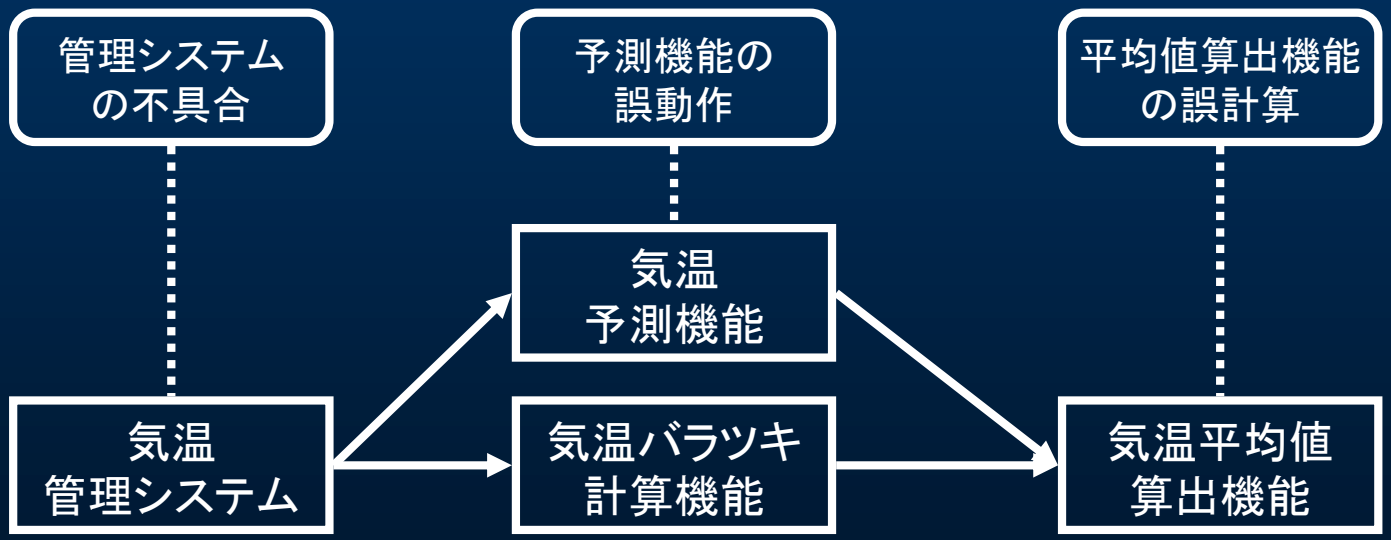
具体例:今回の開発で不具合知識を活用

- 今回の開発するシステム
  - 気温管理システム
    - 気温を予測する機能
    - 気温のバラツキを計算する機能
      - 気温予測機能と気温バラツキ計算機能は気温の平均値を使用する



具体例：今回の開発で不具合知識を活用

- 不具合知識：
  - 管理システムに不具合があった
  - 予測機能が誤予測を起こす
  - 平均値算出機能が誤計算を起こす



具体例：知識の抽出と活用の検討

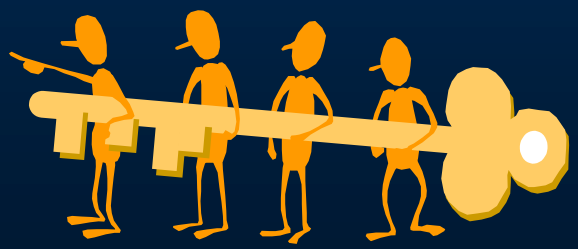
- 過去の不具合の根本的原因
  - 各月の降水量を格納しているデータ構造が例外的だった
    - 6月と9月は降水量が多いので警報がすぐに出せるように
      アクセスが早い別のところに格納していた
      →6月と9月のデータが抜けていた
      - 正しく降水量の合計ができていなかった
- 活用結果
  - 気温のデータ構造は通常通り1月～12月まで並んでいた
    - 的確に不具合が予測できなかった

プレゼンテーションの流れ

- 研究の背景
- 現状の問題点
- 提案内容
- 適用例
- まとめ

ソフトウェアで抽出すべき不具合知識は？

- 「機能」に着目して不具合知識を抽出しても
  的確な不具合の予測はできなそうである
  - ソフトウェアの「機能」は
    ハードウェアの「部品」に近い考え方だろう
- ソフトウェアは高度な知的生産物である
  - 本質的な不具合知識には
    人の側面が必要不可欠だろう
  - 開発者が設計を行う過程で
    どこかに不具合が起こっている



対象とする不具合の典型例

- よく見られる不具合
  - このような不具合をどうにかしたい

仕様

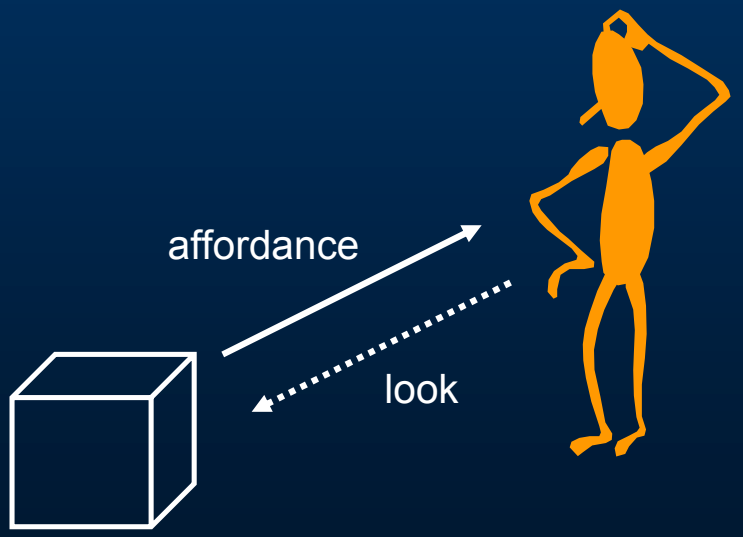
ノーマル状態	通信状態	Menu表示状態	省電力状態	再生中状態
メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示	メニューSW 押下で 前の画面へ	メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示

結果

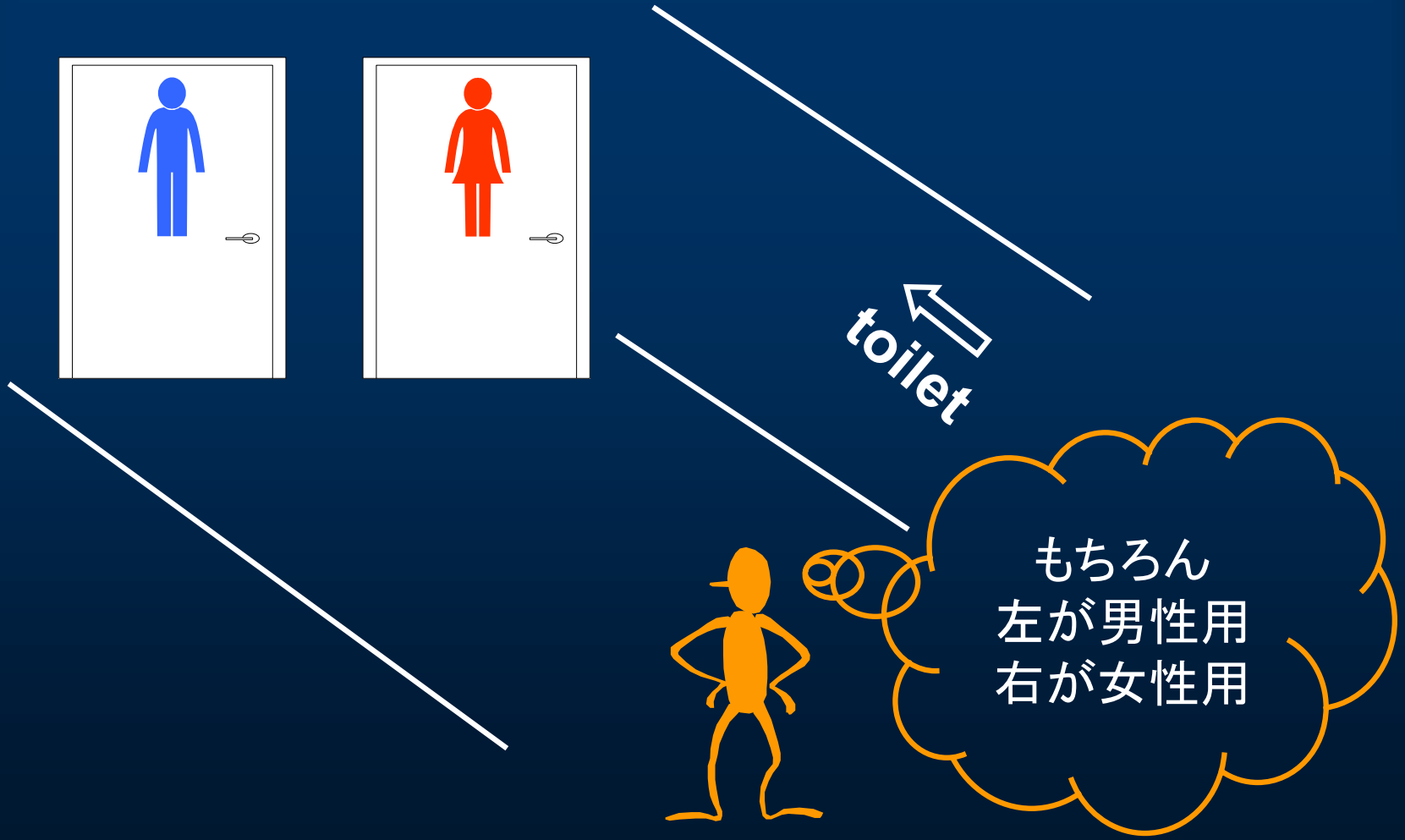
ノーマル状態	通信状態	Menu表示状態	省電力状態	再生中状態
メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示

アフォーダンスを応用する

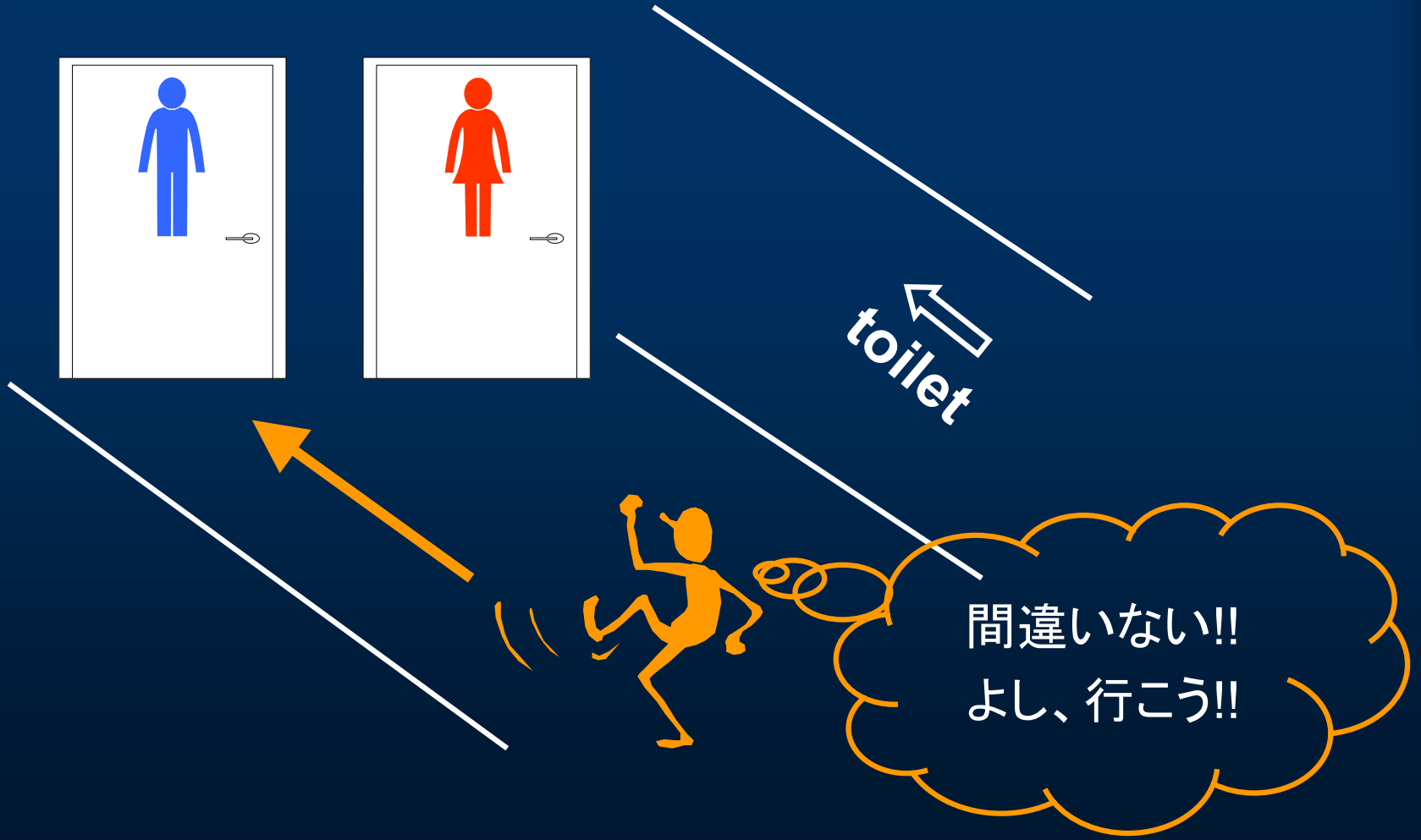
- ロジカル･アフォーダンス
  - 認知心理学で提案されている「アフォーダンス」をソフトウェア開発に応用する
- アフォーダンスとは?
  - モノをどのように使うのかを決めさせるそのモノの属性



アフォーダンスの良い例

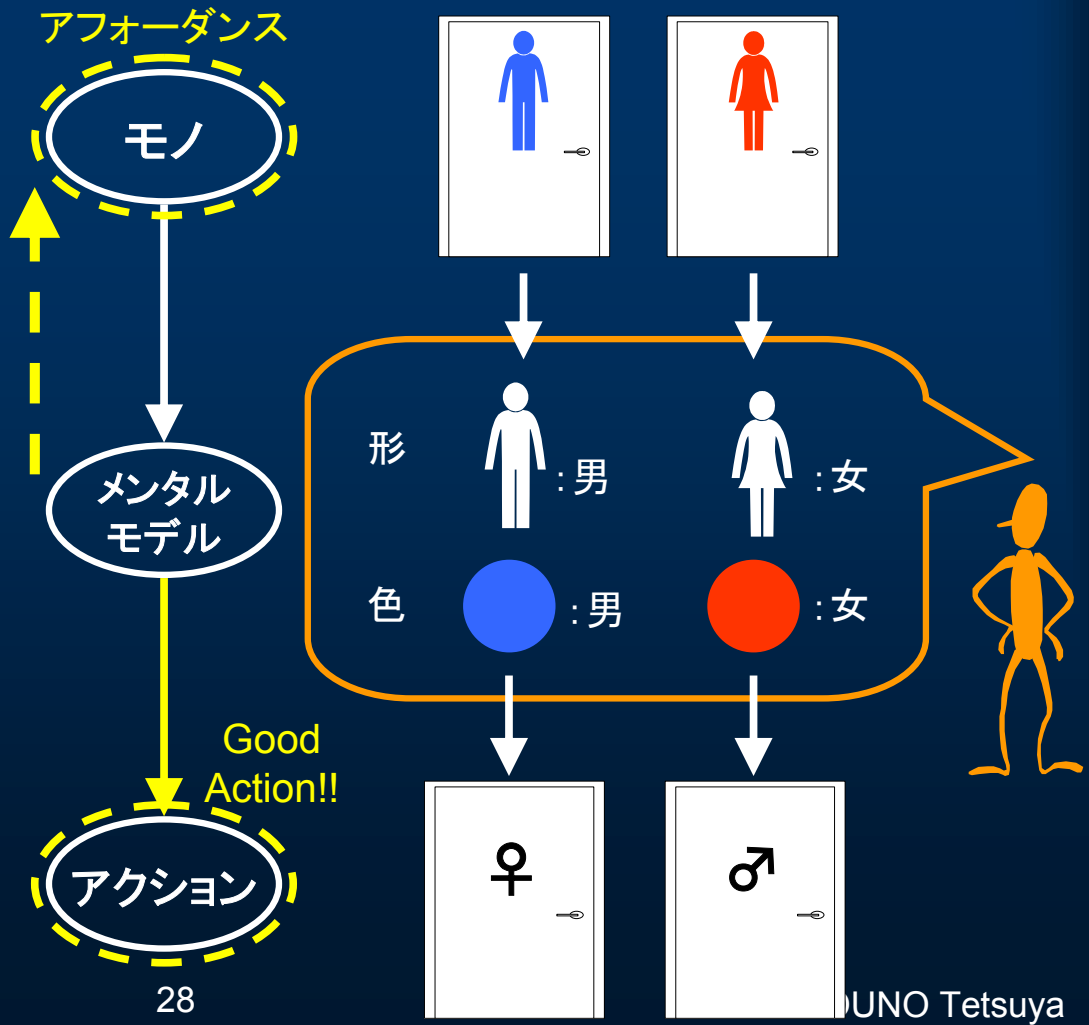


アフォーダンスの良い例

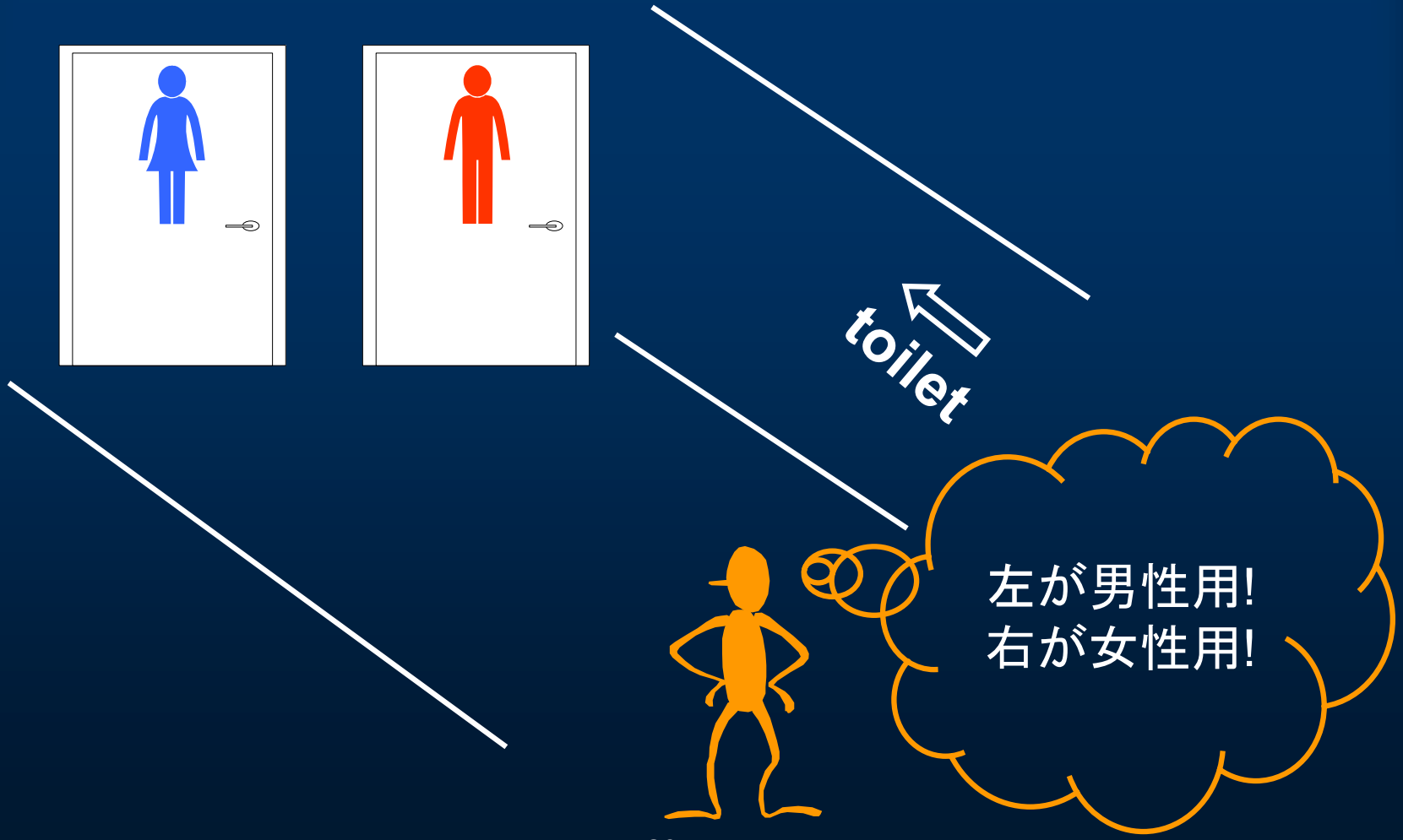


良いアフォーダンスの考察

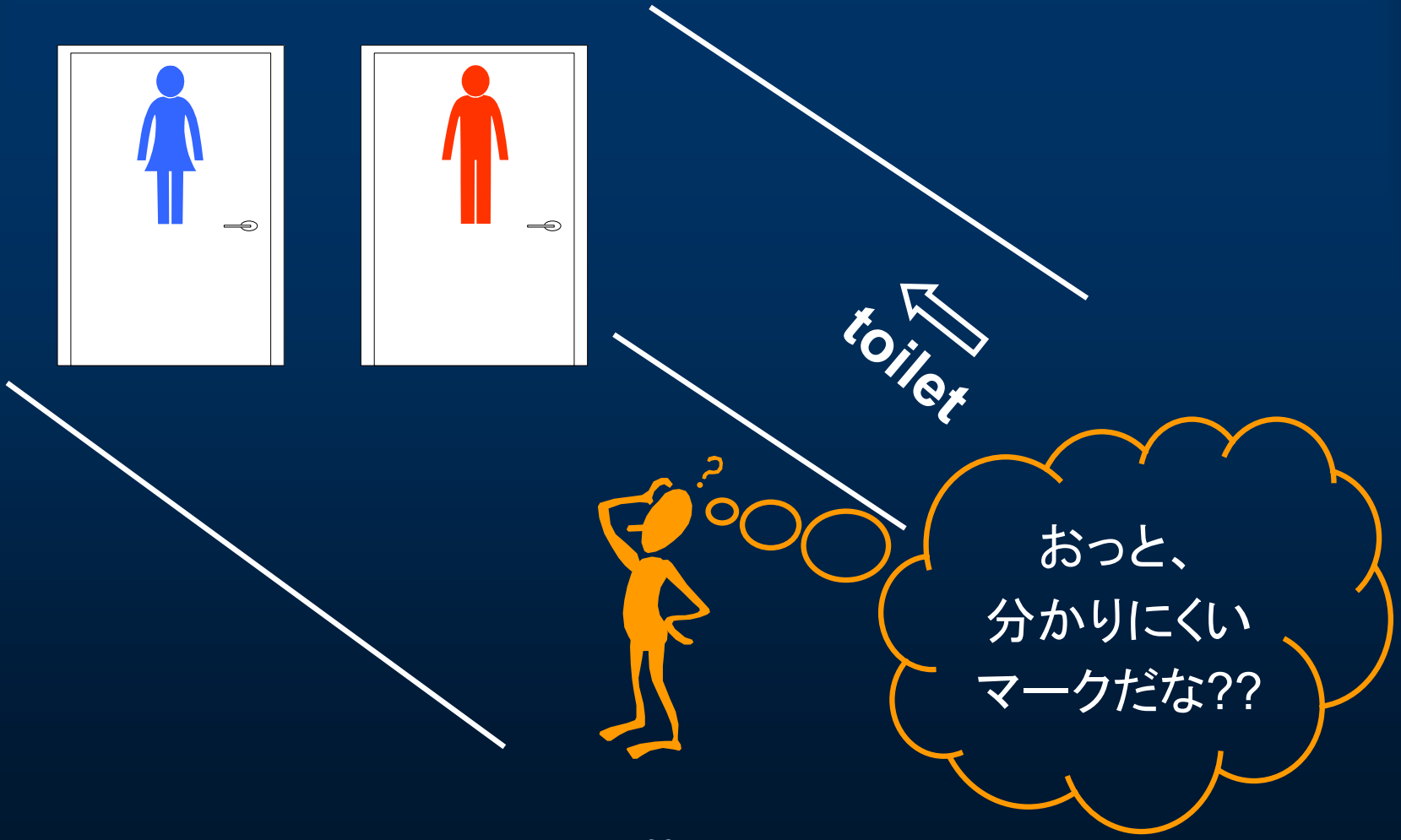
- なぜ男性用と女性用が分かるのか?
  - 我々が持つメンタルモデルを利用しているからである
- アフォーダンスとは
  - メンタルモデルの側面からみたモノの属性である
    - 良いアフォーダンスはメンタルモデルとモノのデザインが一致している



アフォーダンスの悪い例

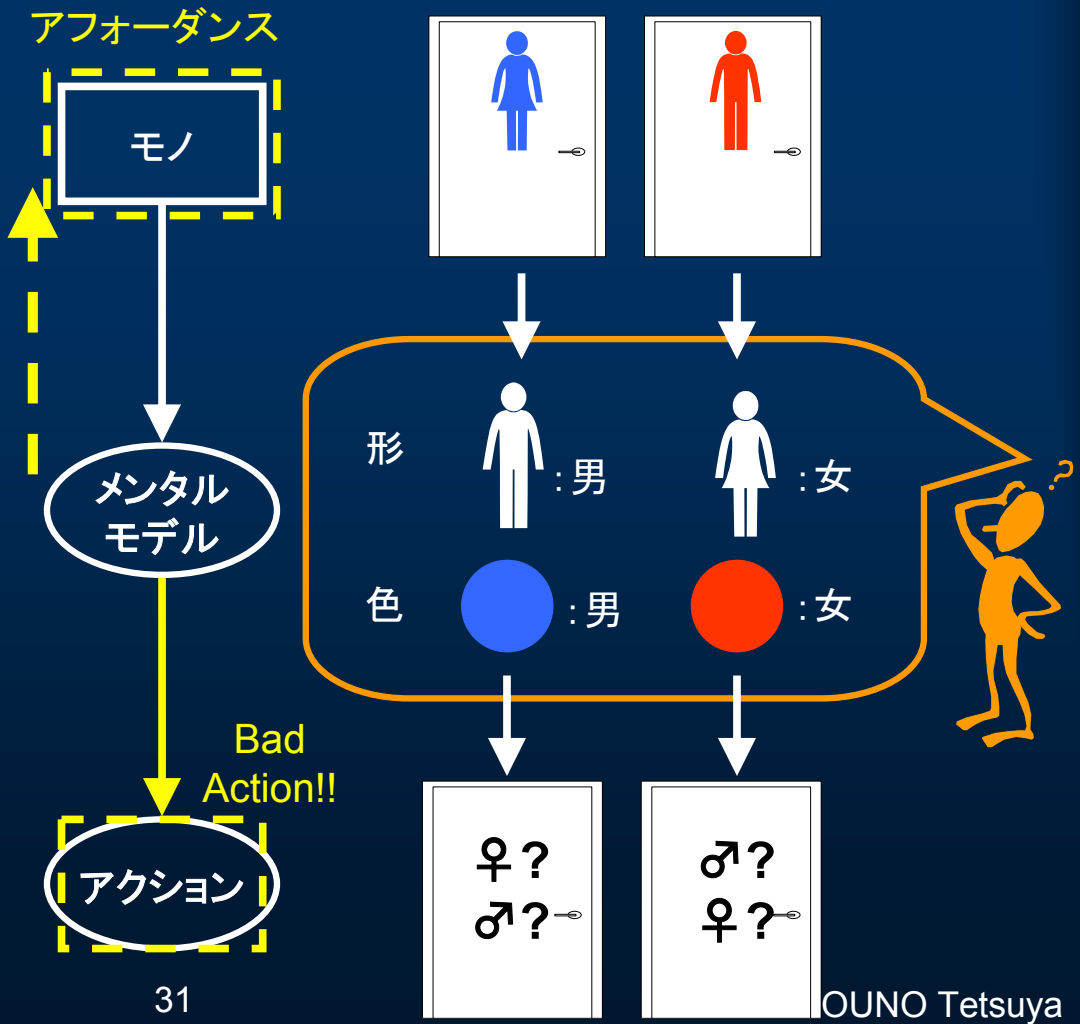


アフォーダンスの悪い例

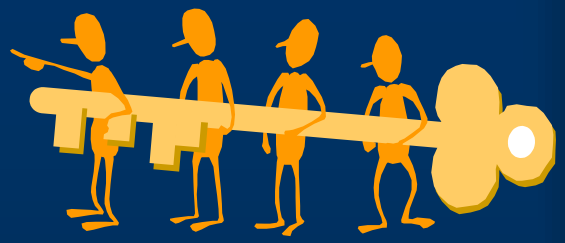


悪いアフォーダンスを少し考察してみよう

- 悪いアフォーダンス
  - メンタルモデルと
    モノのデザインとの
    インタラクションが悪い
  - モノの使い方を
    混乱させてしまう
- 悪いアフォーダンスは
  ヒューマンエラーの
  温床である
  - メンタルモデルとの
    インタラクションが悪い
    モノのデザインを
    整理しておけば、
    ヒューマンエラーを
    防止できそうである



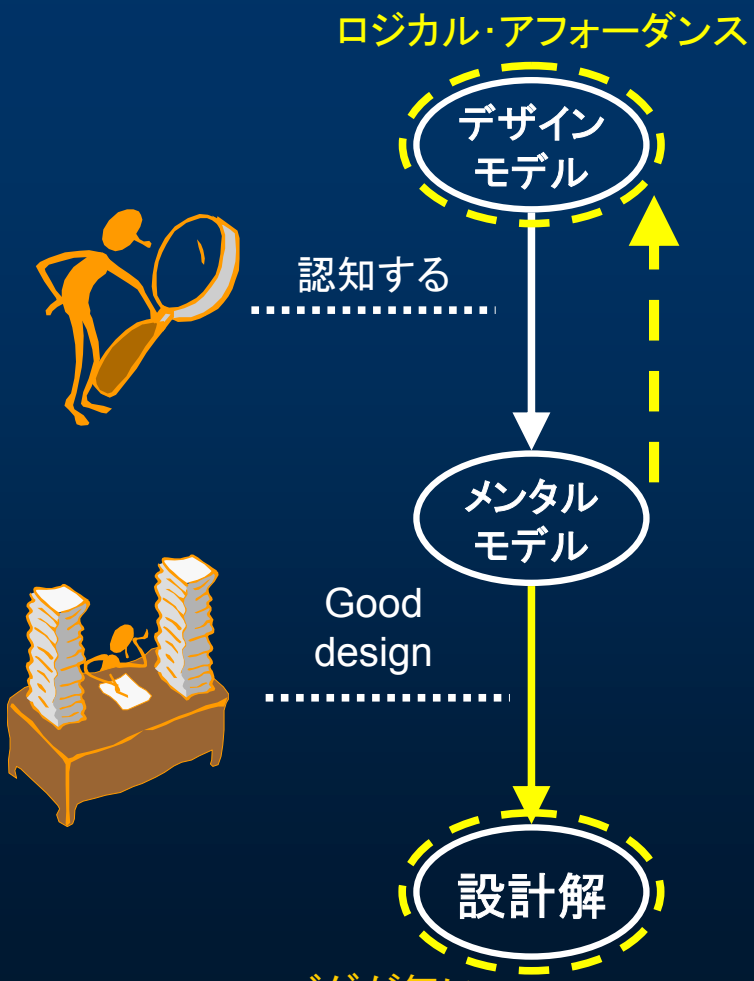
不具合を作り込むメカニズムの明確化の流れ



- 研究のアプローチ
  - 悪いアフォーダンスは
    ヒューマンエラーの温床である
  - このメカニズムはソフトウェア開発にも
    応用できそうである
- 不具合を作り込むメカニズムを明らかにする
  - ロジカル･アフォーダンスを提案する
    - 不具合知識として不具合モードを提案する
  - ロジカル･アフォーダンスに基づき
    不具合を作り込むメカニズムを明らかにする

ロジカル･アフォーダンス

- ロジカル･アフォーダンスとは?
  - 開発者に蓄積されている
    メンタルモデルと
    設計要件の持つモデルとの
    相互作用によって
    どのような設計にすべきかを
    決めさせる設計要件の属性

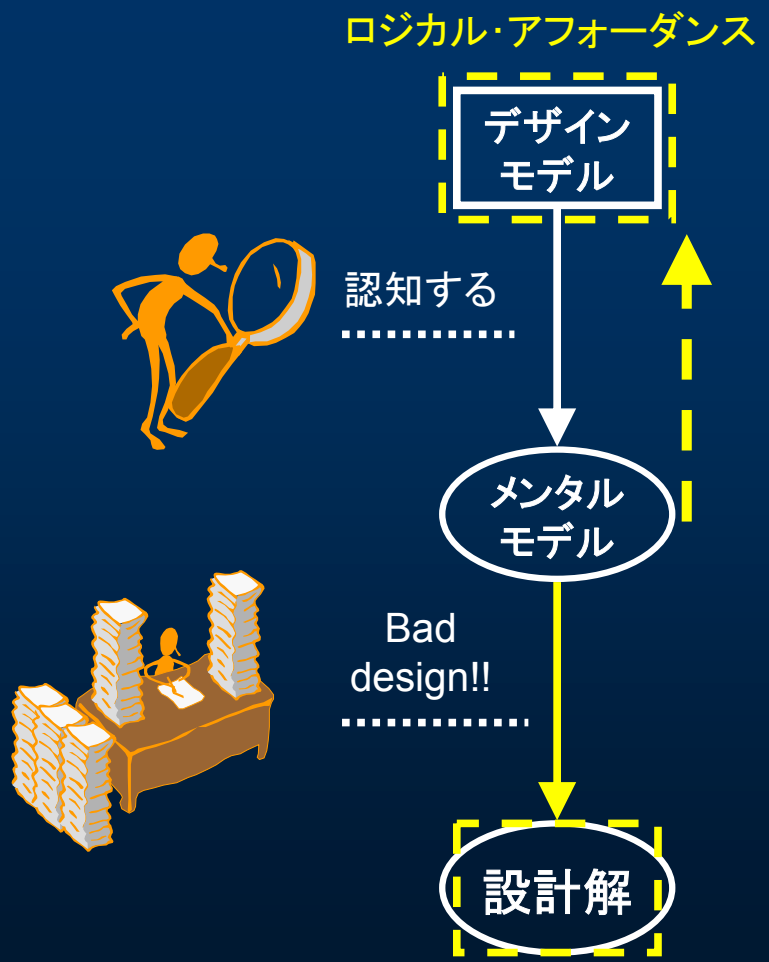


メンタルモデル/デザインモデル

- デザインモデルとは?
  - 設計要件が持つモデル
  - 自然言語としての意味的なモデルや構造としての図形的なモデル
    - 例:「警告文は赤で表示する」 ⇒ 警告文＝赤
    - 例:「入力値を大きい順に並べる」 ⇒ 配列 or ツリー
      - UML、DFDなどのモデルそのものではない
- メンタルモデルとは?
  - 経験や常識などから構築された頭の中のモデル
  - 意味的なモデルや視覚イメージ的なモデル
    - 例:「“赤”は危険、女性」
    - 例:チェスでは頭の中で駒を移動させて、次の手を考える

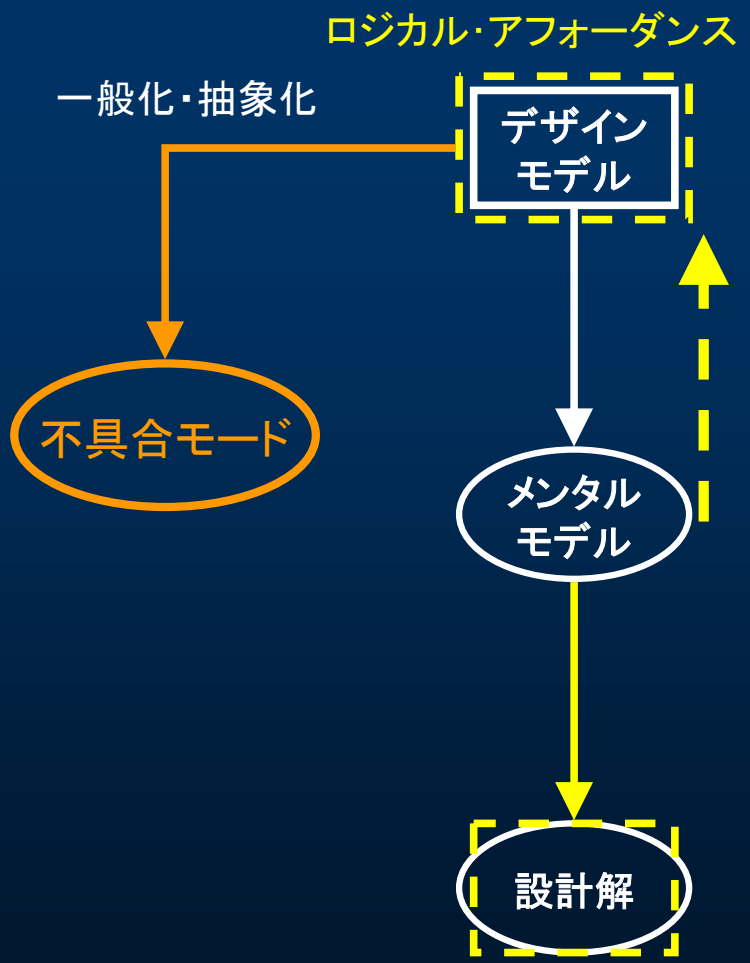
不具合を作り込むメカニズム

- ロジカル･アフォーダンスより
  不具合を作り込むメカニズムを
  明らかにする
  - 開発者に蓄積されている
    メンタルモデルと
    デザインモデルとの
    相互作用の不具合によって
    不具合を作り込む
    - 例えば、
      「警告メッセージを
      青色にする」という
      仕様に対して
      我々が持つメンタルモデルは
      「警告＝赤」であるので
      相互作用の不具合が
      発生してしまい、
      不具合を作り込んでしまう



不具合知識：不具合モード

- 不具合モード
  - 相互作用の不具合が発生してしまう
    デザインモデルを
    一般化・抽象化したものを
    不具合モードと呼ぶ
  - 不具合を作り込みやすい
    設計要件のパターンともいえる
    - メンタルモデルとの
      インタラクションが悪い
      設計要件のデザインを
      整理しておけば、
      ヒューマンエラーを
      防止できそうである
- 不具合を分析することにより
  抽出しなければならない
  不具合知識は
  「不具合モード」である



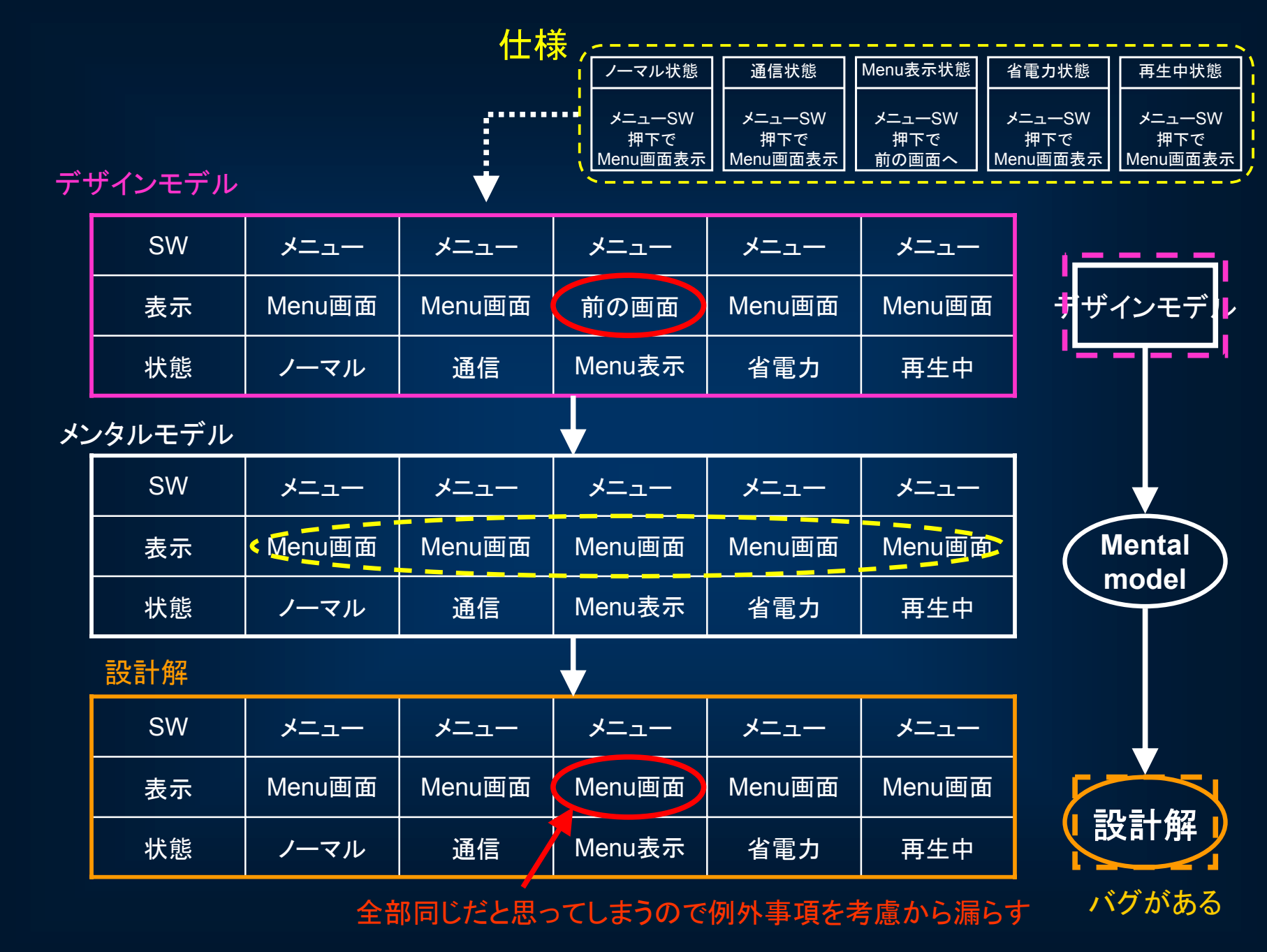
具体例

- 不具合モード「例外」
  - 同じような要素に例外的な要素がごく少数だけ含まれているような仕様のパターン
- 開発者が構築するメンタルモデル
  - 仕様や設計に同じような要素が点在する場合はその要素を一つの集合体としてグルーピングを行う

仕様

ノーマル状態	通信状態	Menu表示状態	省電力状態	再生中状態
メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示	メニューSW 押下で 前の画面へ	メニューSW 押下で Menu画面表示	メニューSW 押下で Menu画面表示

Menu表示状態でメニューSW押下で前の画面へ遷移が「例外」


仕様
ノーマル状態
メニューSW 押下で Menu画面表示
通信状態
メニューSW 押下で Menu画面表示
Menu表示状態
メニューSW 押下で 前の画面へ
省電力状態
メニューSW 押下で Menu画面表示
再生中状態
メニューSW 押下で Menu画面表示
デザインモデル
SW メニュー メニュー メニュー メニュー メニュー
表示 Menu画面 Menu画面 前の画面 Menu画面 Menu画面
状態 ノーマル 通信 Menu表示 省電力 再生中
メンタルモデル
SW メニュー メニュー メニュー メニュー メニュー
表示 Menu画面 Menu画面 Menu画面 Menu画面 Menu画面
状態 ノーマル 通信 Menu表示 省電力 再生中
設計解
SW メニュー メニュー メニュー メニュー メニュー
表示 Menu画面 Menu画面 Menu画面 Menu画面 Menu画面
状態 ノーマル 通信 Menu表示 省電力 再生中
全部同じだと思ってしまうので例外事項を考慮から漏らす
デザインモデル
Mental model
設計解
バグがある

提案の整理

- 不具合モード:
  不具合を作り込みやすそうな設計要件のパターン
  - メンタルモデルの側面から捉えた
    ヒューマンエラーを誘発し易い設計要件のパターン
  - メンタルモデルとの相互作用に
    不具合が起きやすそうな設計要件
- 不具合を分析することで
  不具合を作り込みやすそうな
  設計要件のパターンを抽出する
  - このような視点で分析すると
    「うっかりしていた」といった分析結果にならないだろう

プレゼンテーションの流れ

- 研究の背景
- 現状の問題点
- 提案内容
- 適用例
- まとめ

適用例：不具合モードをレビューに活用した

- 適用対象
  - 「例外」を抽出した不具合とは違う組織でかつ違う機能の仕様
    - 状態遷移表
- 状態遷移表のレビューに「例外」を活用した
  - 「例外」に当てはまっているかどうか分かり易くするために状態遷移表を整理した
  - 整理した成果物より、5件が例外に当てはまった
  - その5件をレビューで確認する項目とした
- 適用結果
  - 5件のうち、2件が仕様の不具合であった
    - そもそもその例外項目が間違いであった
      - 「例外」に当てはまる仕様は、仕様の不具合の可能性もある
  - 残りの3件はまだ調査できていない

プレゼンテーションの流れ

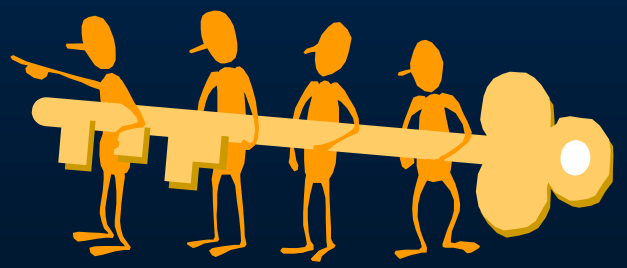
- 研究の背景
- 現状の問題点
- 提案内容
- 適用例
- まとめ

まとめ

- 不具合を分析し、メカニズムが明らかになれば市場での不具合を減らすことができる
  - 不具合に関する本質的な知識を抽出する必要がある
    - 「機能」に着目しても本質的な知識は抽出できなそうである
- ロジカル･アフォーダンスの提案
  - 不具合を作り込むメカニズムを明らかにした
  - メンタルモデルとのインタラクションが悪い設計要件のデザイン、すなわち不具合モードを整理しておけば、不具合の作り込みの未然防止、不具合の検出に活用できるのではないか

今後の課題

- メンタルモデルおよびデザインモデルの詳細な検討、整理
- 多くの適用による不具合モードの拡充
- 設計レビューで不具合モードを活用するための取り組み
- 不具合の作り込みの未然防止で不具合モードを
- 活用するための取り組み



ご清聴ありがとうございました

電気通信大学大学院
河野 哲也
kouno@se.uec.ac.jp