JAIST Repository
https://dspace.jaist.ac.jp/

Title	モデル検査によるアーキテクチャ設計検証
Author(s)	岸，知二
Citation	
Issue Date	2006-11-28
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8314
Rights	
Description	3rd VERITE : JAIST/TRUST-AIST/CVS joint workshop on VERIfication TEchnologyでの発表資料，開催：2006年11月27日～28日，開催場所：JAIST 知識科学研究科講義棟・中講義室

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

モデル検査による アーキテクチャ設計検証

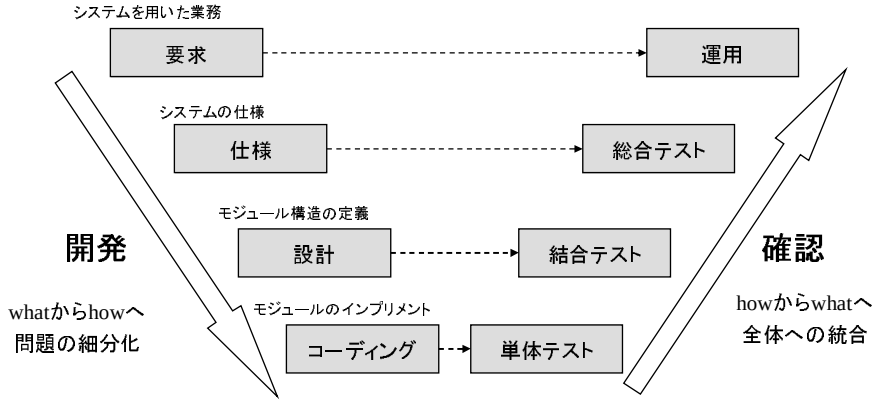
2005/11/28

岸知二(JAIST)

設計の重要性

- 多くの問題が上流工程に起因
  - 手戻りコスト
- 組み込み分野の品質問題
  - 産業界も設計品質に大きな問題意識
- 設計とはアーキテクチャ設計を意味
  - 詳細設計は除外
  - 上位の設計意図が考慮の対象

開発のV字型モデル



※いろいろな名称や分類があります。

ソフトウェアアーキテクチャとは

- ソフトウェアおよびその開発を支配するソフトウェア構造や構造化の原則
  - ソフトウェアの諸特性を決定付ける
  - 作業分担や作業の手順など様々な開発に多大な影響を持つ
  - 変更の影響が多大
  - 構造決定に対する様々な原則をも含む

アーキテクチャ設計の特徴

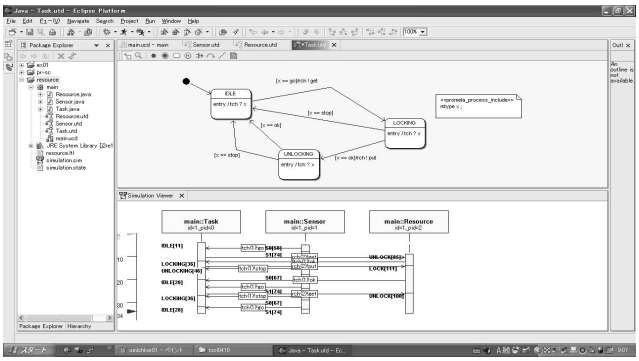
- 骨格構造の設計
  - 目的指定から抽象化され必要十分な範囲を記述
- シナリオを活用した評価
  - 重要・特徴的な状況に照らして妥当性を検討
- アーキテクチャ上の想定に対する考慮
  - 特定のシステムやプラットフォーム上での確認ではなく、アーキテクチャとしての広がりを想定

モデル検査技術

- 有限状態モデル上で論理的な性質が成立するかどうかを自動的に検査
- ソフトウェア検証への期待
  - 網羅的な検査によりテスト以上に確実な検証が行える
  - 並行動作、イベントの順序・タイミング・競合状況などの網羅的な検査が可能
- アーキテクチャ設計検証へ適用できるか？

モデル検査による設計検証ツール

- UML設計のSPINでの検証を支援



適用上の課題 (1/2)

- 設計モデルの厳密性・詳細度
  - 通常の設計モデルと、モデル検査のための検証モデルとは大きな違い
    - 設計モデル：人間が読むための記述、動的な意味の扱いも不正確
    - 検証モデル：厳密、詳細、ナイーブ
  - 現実規模のシステム全体を、検証モデルの精密さで記述することは困難

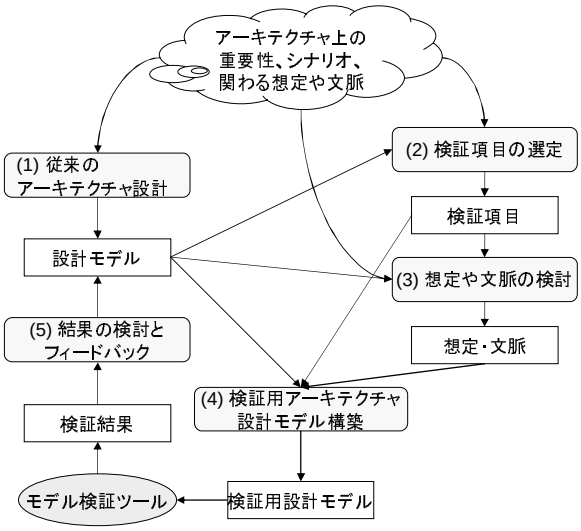
適用上の課題 (2/2)

- 検証項目の選定
  - 検証可能な事項は大量に考えられる
    - 様々な機能、様々なインスタンス構造、..
    - 設計検証はテストではない
  - 設計段階において何を検証することが有効か
- 網羅検査適用の妥当性
  - アーキテクチャ検証は一般に発見的
  - モデル検査が活きる検証とは何か

アプローチ

- 精密さ→アーキテクチャ上の重要性に関わる観点からの抽象化
  - アーキテクチャ上の重要性に照らして、抽象化、記述範囲の限定を行う
- 項目→重要シナリオに照らした項目の選定
  - アーキテクチャ上重要なシナリオを確認
- 網羅検証→想定に関する網羅検証
  - 想定されるコンテキストに関する十分な

適用手法の提案



想定モデルとは

- 検証対象の使われ方の想定を示すモデル
  - 典型的にはプロダクトラインのコア資産を想定
  - 暗黙的な想定を明示化するモデル
- 検証対象と環境を含むモデルから導出
- 留意点:
  - 想定は元々は開発者の意図に基づく
  - 現時点ではシステム基盤に関する想定に関しては必ずしも表現しきれていない

想定モデルのねらい

- アーキテクチャ上の想定を明示化する
  - 必要かつ十分な検証内容を検討
- 現実的な検証量となる利用方法を検討
  - 導出されうるすべての可能性を検証することは非現実的
- 検証が容易な設計を行う
  - 設計段階から妥当な検証量を意識する

想定モデルの導出方法

1. コア資産の範囲の明確化
2. コア資産を利用するプロダクト群の明確化
3. そのプロダクト群から導出される設計モデルを明確にする
4. 設計モデルから想定モデルを作成
   1. 導出設計モデルに含まれない要素の除去
   2. 導出可能な範囲で制約を強める

得られる想定モデル

- 直感的には:
  - プロダクトラインのモデルから不要なモデル要素を除去し、より制約を強化したもの
- 具体的には:

ユースケース	システムと環境のかかわりに関する想定（使われ方等）
クラス図 ステート図	コア資産と他との関係に関する想定

想定モデルの例 (1/2)

- 以下のP0, P1, P2がコア資産を使うと想定

製品	フィーチャ	クラス
P0	標準LCD	CDCtrl-S TunerCtrl-S DspCtrl-S
P1	大画面LCD 圧縮オーディオ再生	CDCtrl-H TunerCtrl-H DispCtrl-H
P2	大画面LCD 圧縮オーディオ再生 交通情報受信	CDCtrl-H TunerCtrl-H DispCtrl-H

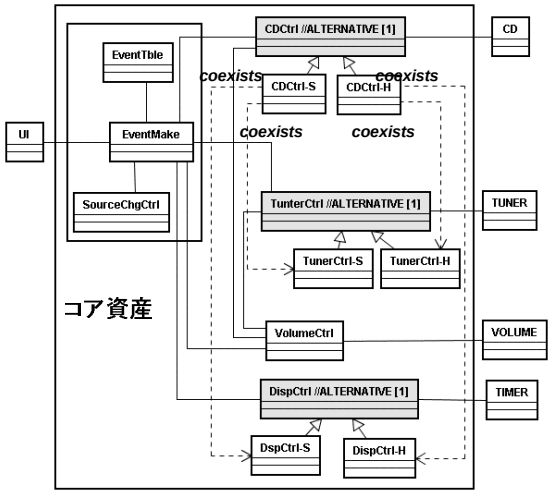
想定モデルの例 (2/2)

カーオーディオの例

以下をコア資産とする
“EventTbl”
“EventMake”
“SourceChgCtrl”

前述の手続きを適用したクラス図

1) 不要な要素を除去
“CDChangeCtrl”
“CDCHANGER”
2) 制約により要素の組み合わせを限定



想定モデルの設計検証への活用

- 外部環境のモデル化
  - 想定モデルのユースケースの活用
  - 送られるイベントを決定
- インスタンス構造の検討
  - 検査を適用する際のインスタンス構造の決定
- 他の内部処理とのかかわり
  - 同時に走るタスクや割り込みのモデル化の同定

議論

- 想定を明示的にモデル化
  - コア資産を必要十分なコンテキストで検証するための参考
- 検証すべきケースは爆発しうる
  - コア資産の利用方法の検討
  - 設計へのフィードバックが本質
- システム基盤への想定の扱いは今後の課題