Collaborative Modelling and Co-simulation

Tools and Techniques for Designing Embedded Systems

John Fitzgerald
Fuyuki Ishikawa
Peter Gorm Larsen

AARHUS
UNIVERSITY

Programme

10:00 – 11:30: Introduction to DESTECS/Crescendo, Co-modelling and Co-simulation
11:30 – 12:30: Quick Introduction to VDM
12:30 – 13:30: Lunch
13:30 – 15:30: Practical Trial with Line-following Robot
15:30 – 16:00: Coffee Break
16:00 – 17:30: Application Experiences, Implications, Discussions

The DESTECS Team

DESTECS

SEVENTH FRAMEWORK PROGRAMME

Design Support and Tools for Embedded Control Systems

(Jan 2010 – Dec 2012), www.destecs.org

John Fitzgerald, Kenneth Pierce, Carl Gamble, Claire Ingram, Peter Gorm Larsen, Kenneth Lausdahl, Augusto Ribeiro, Joey Coleman, Kim Bjerge, Sune Wolff, José Antonio Esparza Isasa, Claus Ballegard Nielsen, Martin Peter Christensen, Jan Broenink, Xiaochen Zhang, Yunyun Ni, Angelika Mader, Jelena Marinčić, Christian Kleijn, Peter Visser, Frank Groen, Marcel Groothuis, Peter van Eijk, Dusko Jovanovic, Jan Remijnse, Eelke Visser, Michiel de Paepe, Koenraad Rombaut, Yoni de Witte, Roeland van Lembergen, Wouter Vleugels, Bert Bos, Jeffrey Simons

UNIVERSITY OF TWENTE. iha.dk Newcastle University VERHAERT chess neopost NEDERLAND

Newcastle University AARHUS UNIVERSITY

Introduction

John Fitzgerald
Peter Gorm Larsen

AARHUS
UNIVERSITY

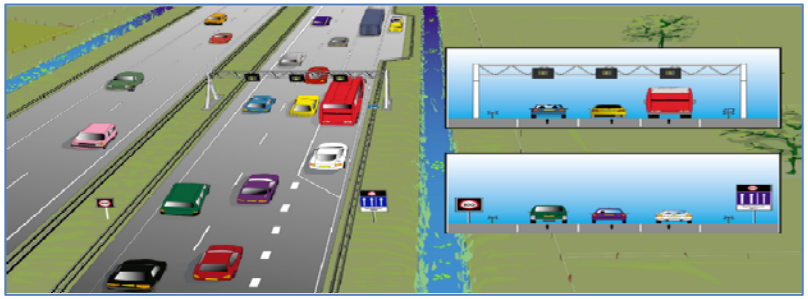
背景

- 計算機はより小さく，高機能に，どこにでも
- ~68億もの携帯電話アカウント
- 例えば自動車：
  - ハイエンドの車種では8個のプロセッサと100M LOCを超えるコード
  - リコールのコストは膨大
    - 3つ以上の問題が起きると，ブランドへの忠誠度は 55% から 39% に

80s

90s

00s

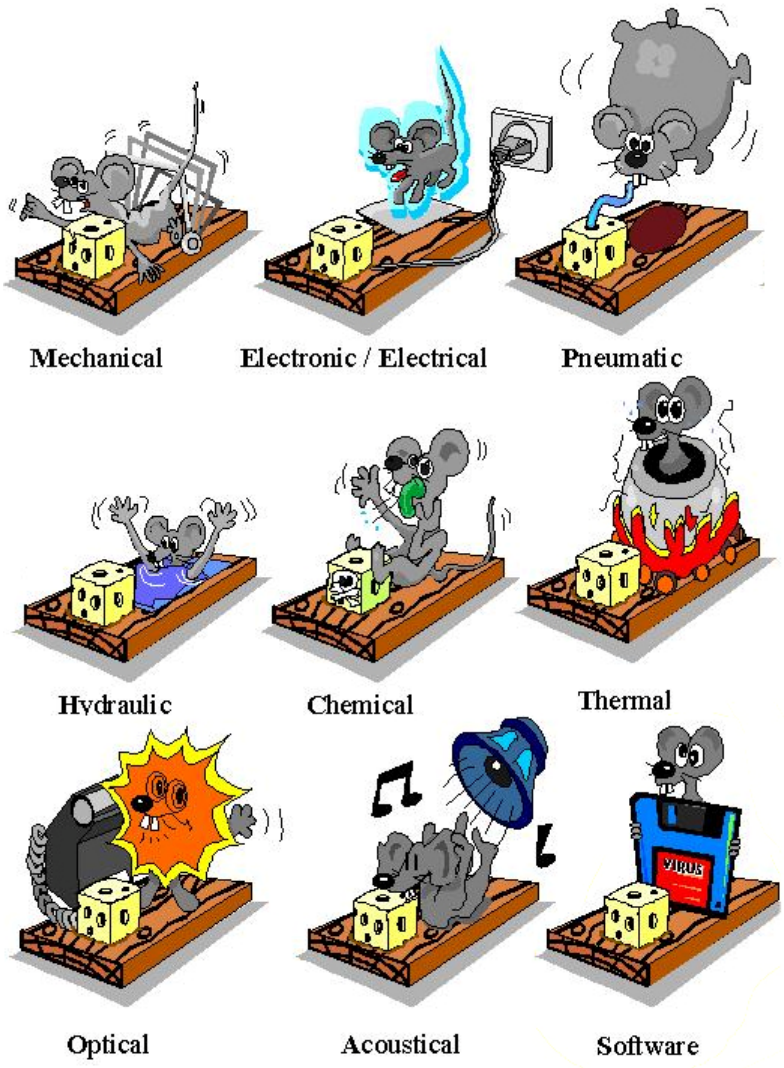


10s

背景

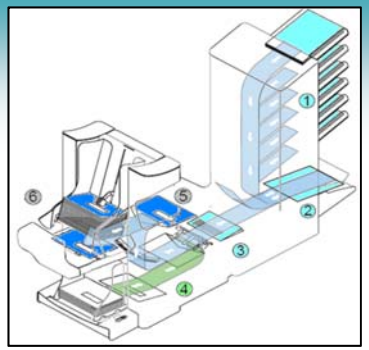

We offer three kinds of service:
GOOD - CHEAP - FAST
You can pick any two
GOOD service CHEAP won't be FAST
GOOD service FAST won't be CHEAP
FAST service CHEAP won't be GOOD
U-NEAK SIGNS

背景



- 各領域への問題の分解
- 市場に送り出す時間の改善のためには並列な作業が必要
- … しかし重要な性質は多領域にまたがる
- … すると問題点が後期になってから（統合時に）発覚することに
- では
  どのように領域間の境界を越える？

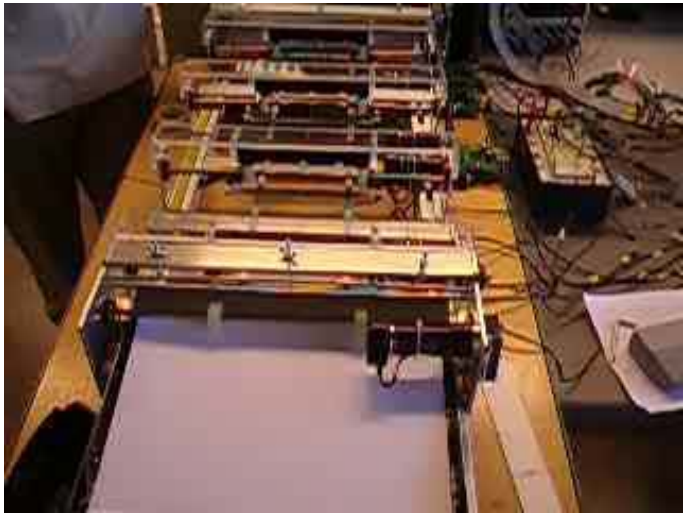
背景：Co-modelling

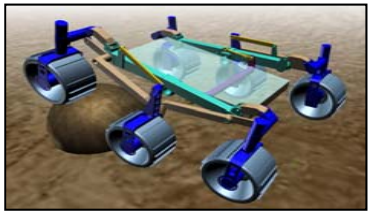


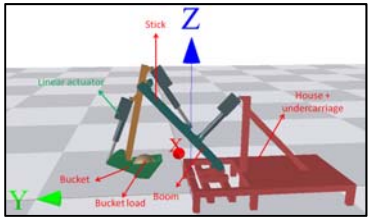
ソフトウェア：
- 連続（Discrete）
- 複雑な論理

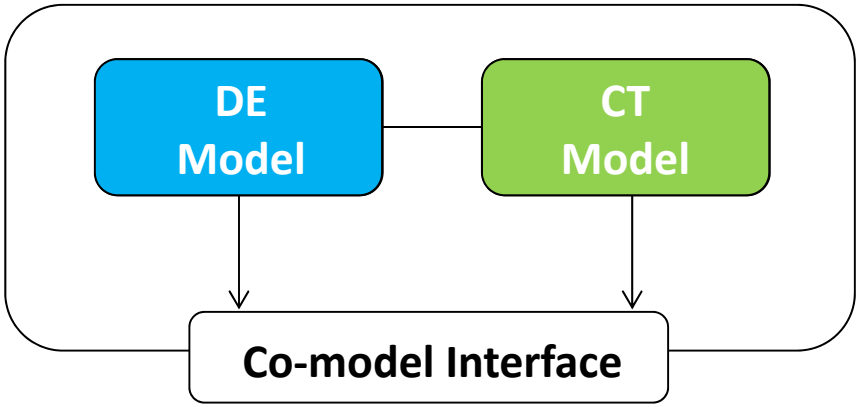
Mind the Gap!

物理:
- 連続（Continuous）
- 数理計算

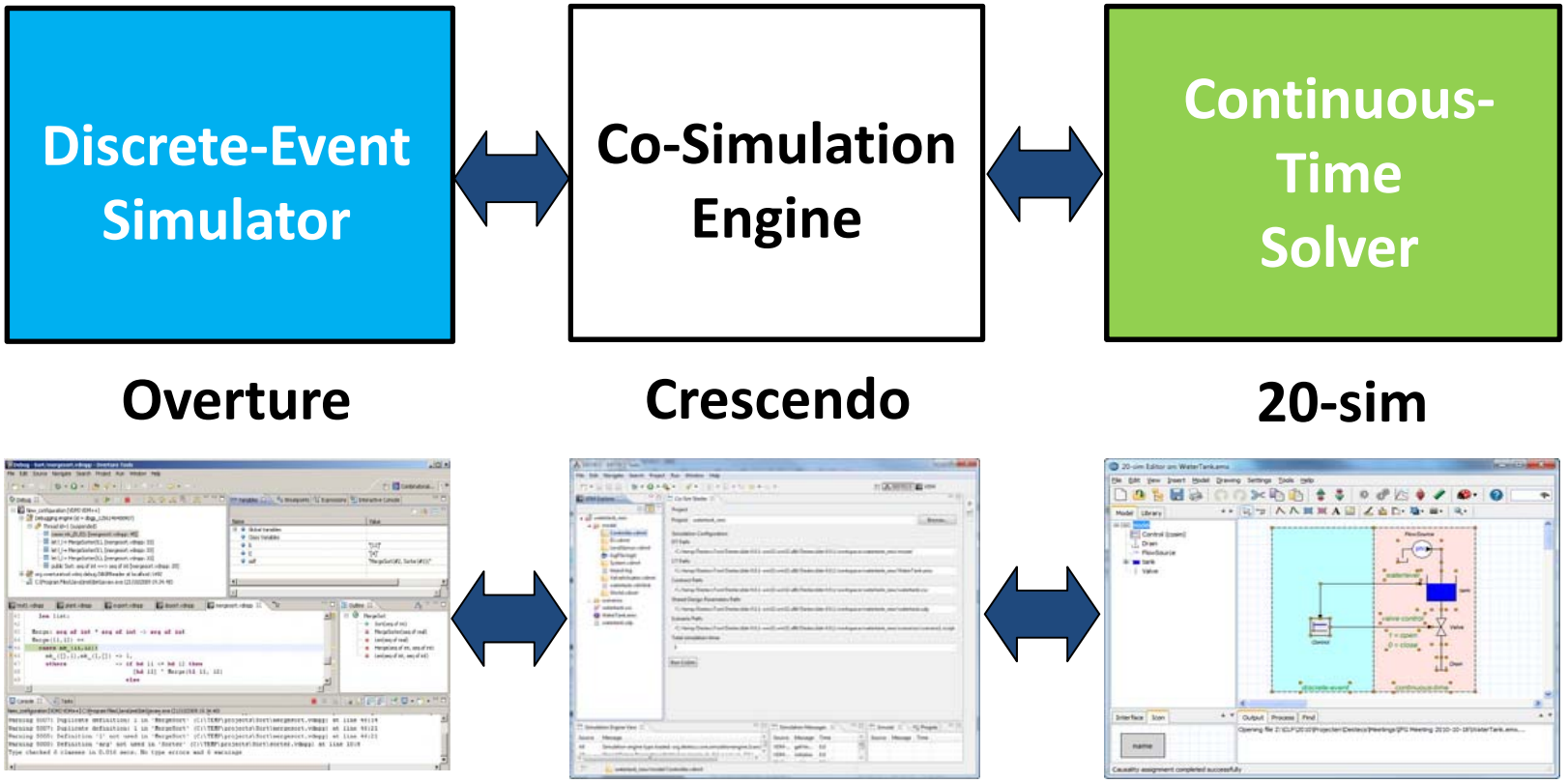




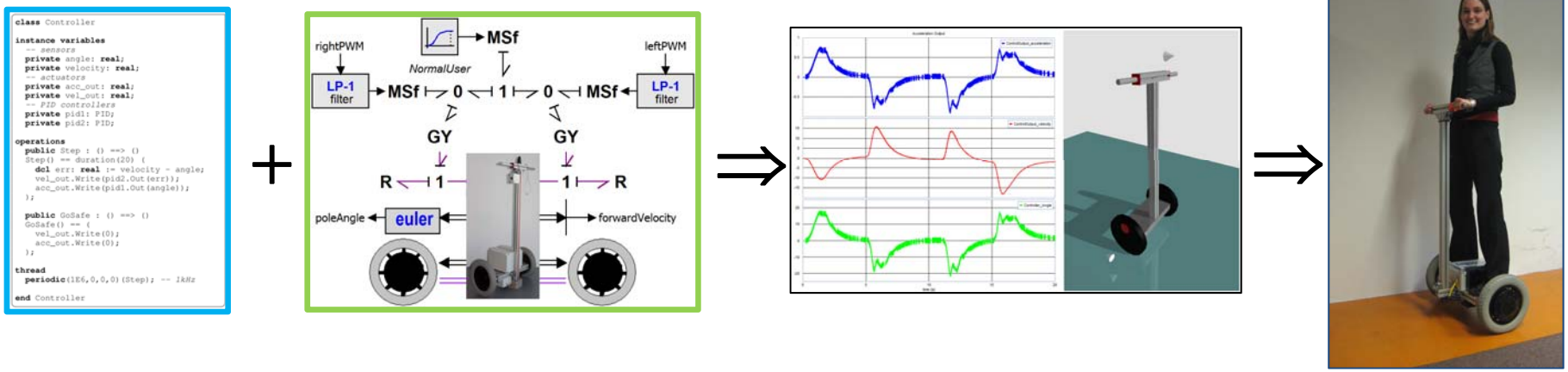




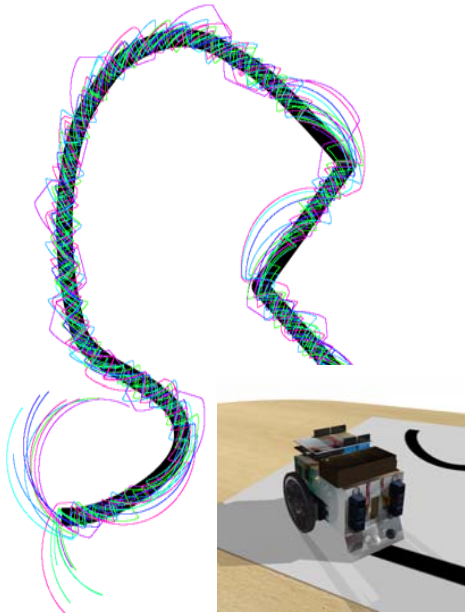
背景：Co-simulation



Co-modelへの支援



- 成果物：ツール（Crescendo），手法，ガイドライン（特に障害のモデリング）
- 自動的なCo-modelの分析（一通りの調査とランキング）
- 設計の反復・コストを交通，機械設計，高速な容姿処理や荷物取り扱いで削減！



DESTECS: Design Support and Tools for Embedded Control Systems

EU FP7 INFSO-ICT-248134 (Jan 2010 – Dec 2012)

Océ	Dutch Space	Vestas
Airbus	ESA	Grundfos
Nokia	FKI Logistex	Volvo
Siemens	Darwind	Bang & Olufsen
Martin Group	ASML	MBDA
Atlas Copco	Assembleon	Terma

Newcastle University

UNIVERSITY OF TWENTE.

controlLab

neopost

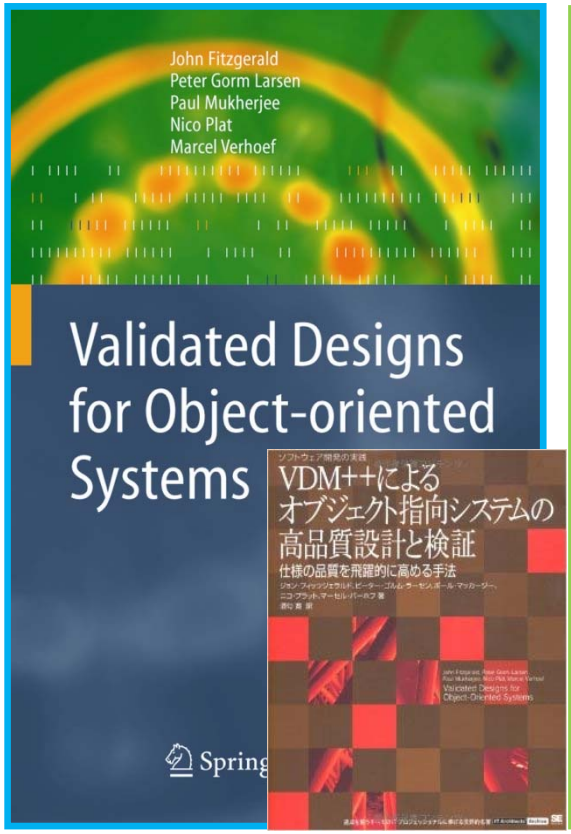
cyber engineering the future

AARHUS UNIVERSITY

VERHAERT

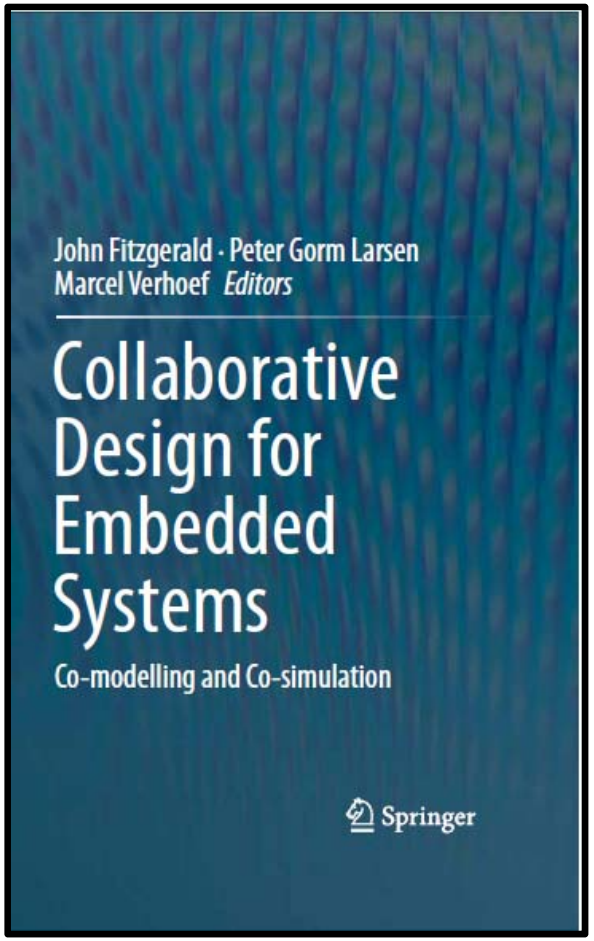
cyber iX

参考書



Baseline Discrete Event Modelling

Baseline Continuous Time Modelling



Co-Modelling

Co-modelling and Co-simulation

John Fitzgerald
Peter Gorm Larsen

モデル駆動の設計

- 近年のシステムはとても複雑
- このため事前のモデル構築による対応が重要
  - 分析を行う(静的解析，証明，モデル検査，シミュレーション)
  - 仮定を明確にする
  - 考えられる設計の可能性を評価する
  - コストの高いプロトタイピングを避ける
- 異なる側面に対して異なるモデリングのパラダイムが存在

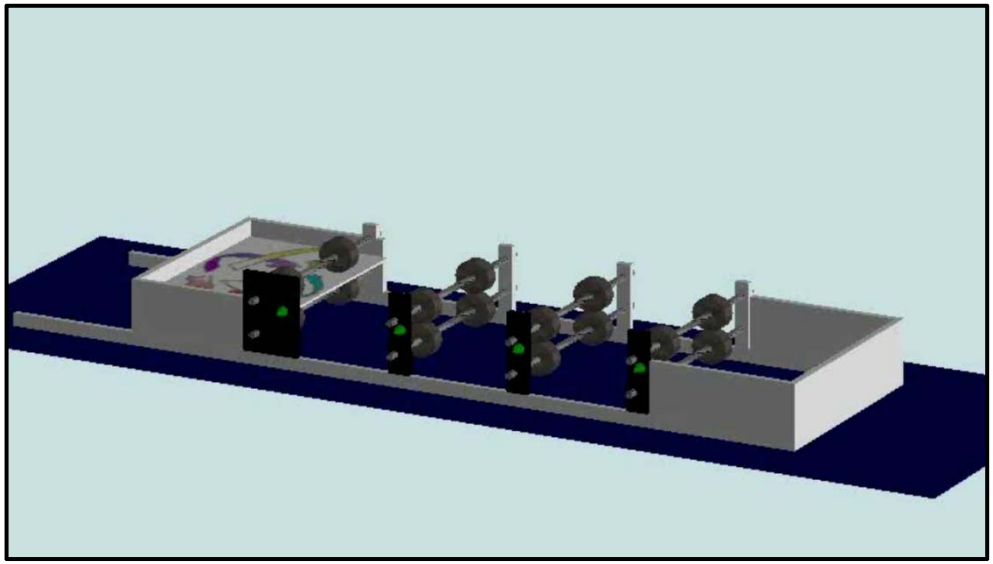
ソフトウェアと物理系のモデリング

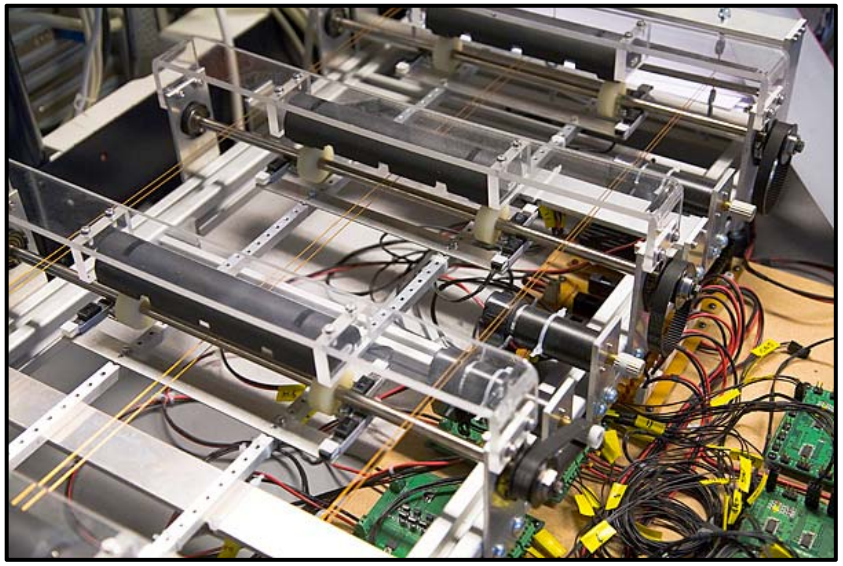
- 典型的には**離散的なイベントモデル**(**discrete-event : DE**), 例えばVDM-RT
- シミュレーションにおいては, 状態変化が起きる時間のみが表現される
- ソフトウェアに対するよい抽象化
  - 例: データ型, オブジェクト指向, スレッド
- 物理的なシステムのモデリングには適していない

- 典型的には連続的な時間モデル(**continuous-time : CT**), 例えば微分方程式
- シミュレーションにおいては, 状態変化は時間経過に伴い連続的に起きる
- 領域ごとの抽象化
  - 例: 機械, 電気, 水力
- ソフトウェアモデリングの支援はわずか
  - 関数やオブジェクトを活用しない基本的なプログラミングの支援

組み込みシステム

- 様々な構成要素（計算，物理，人間）の相互作用
- 制御ソフトウェアの80%にも及ぶ部分について，モード分けなどロジックの複雑さの増加
- エラーの検出と回復
- 協調的な開発
- 多様な領域ごとの文化，抽象化，形式化
- 典型的には個別の取り組み
- **設計空間の探索の必要性**

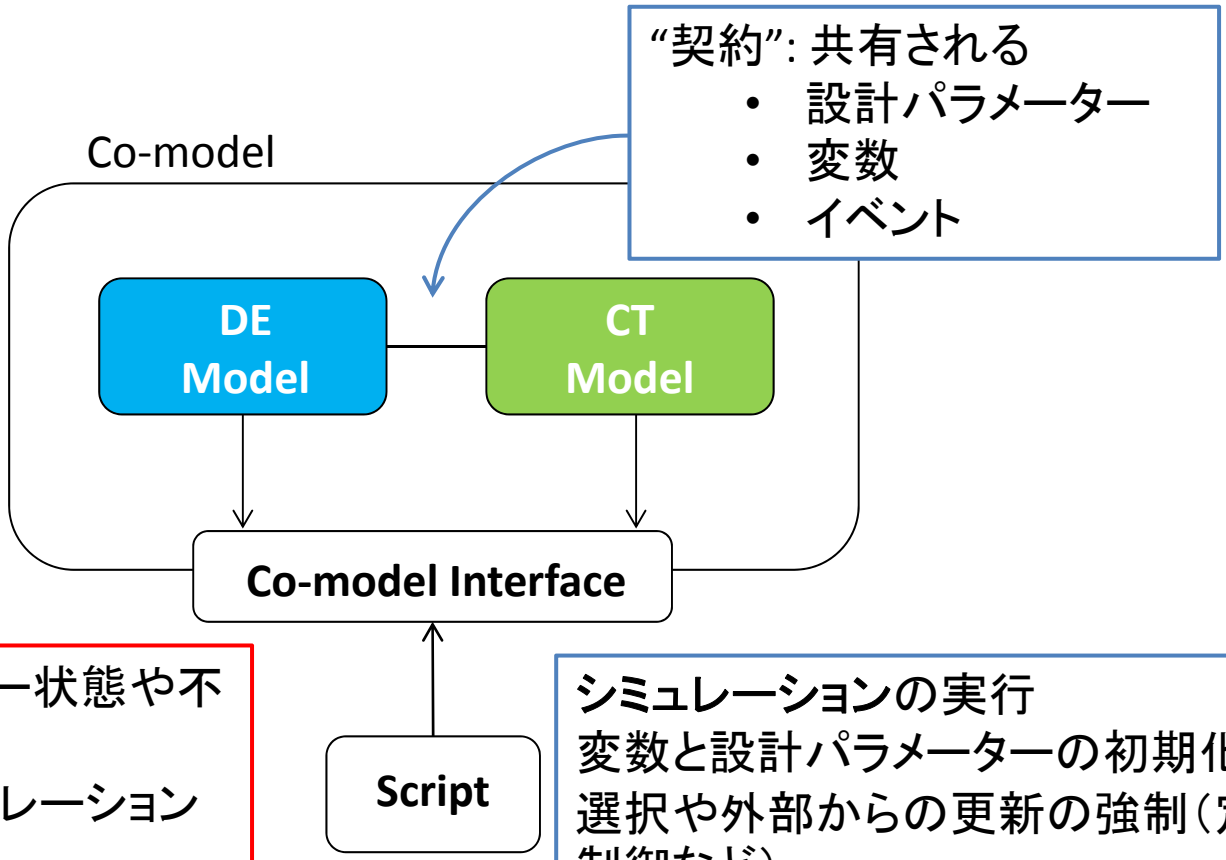




Co-modellingの概念

変数:
実行中に変更

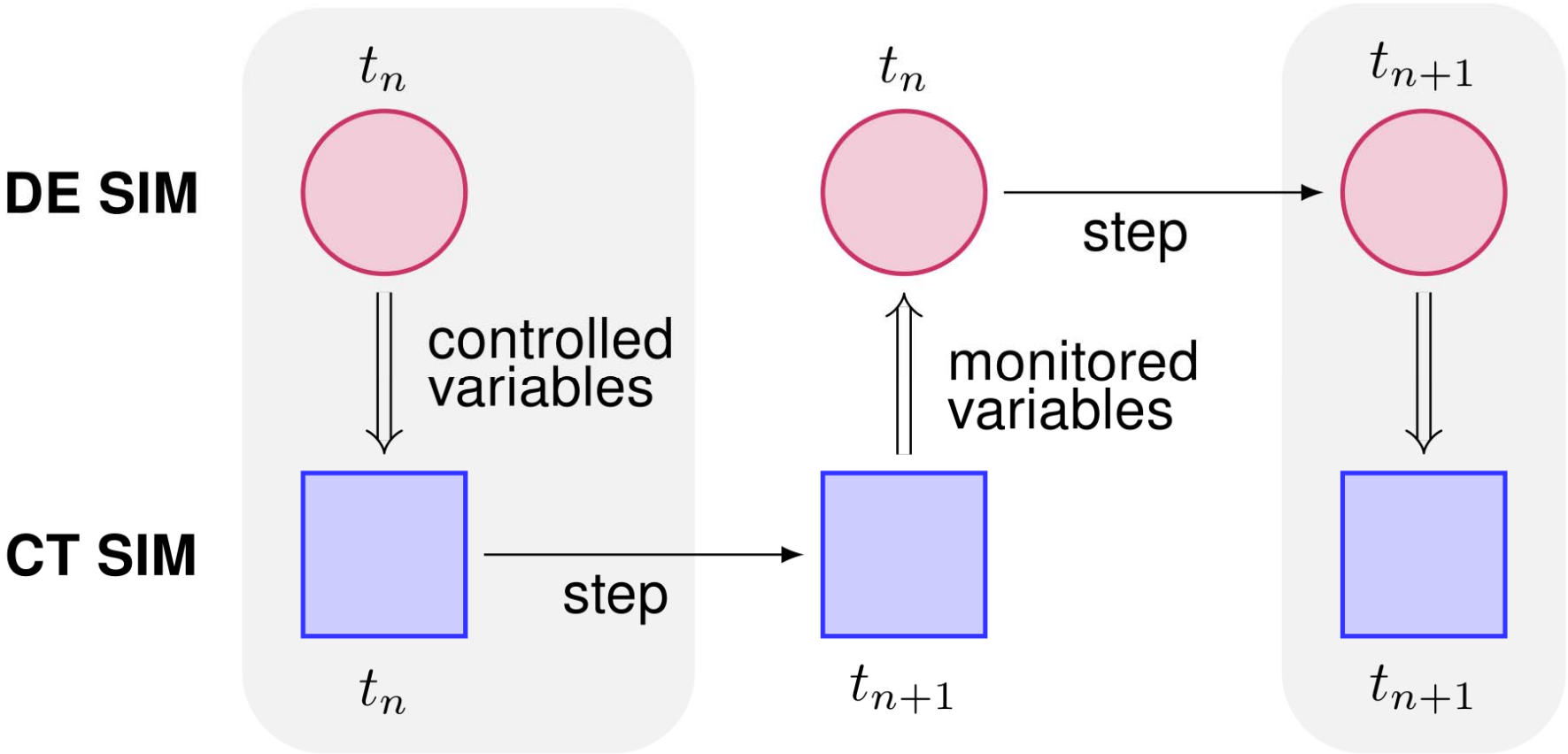
設計パラメーター:
実行中は固定



障害のモデリング:エラー状態や不完全な機能など
障害の埋め込み:シミュレーション中にスクリプトにより

シミュレーションの実行
変数と設計パラメーターの初期化
選択や外部からの更新の強制(定値制御など)

Co-simulationの意味論



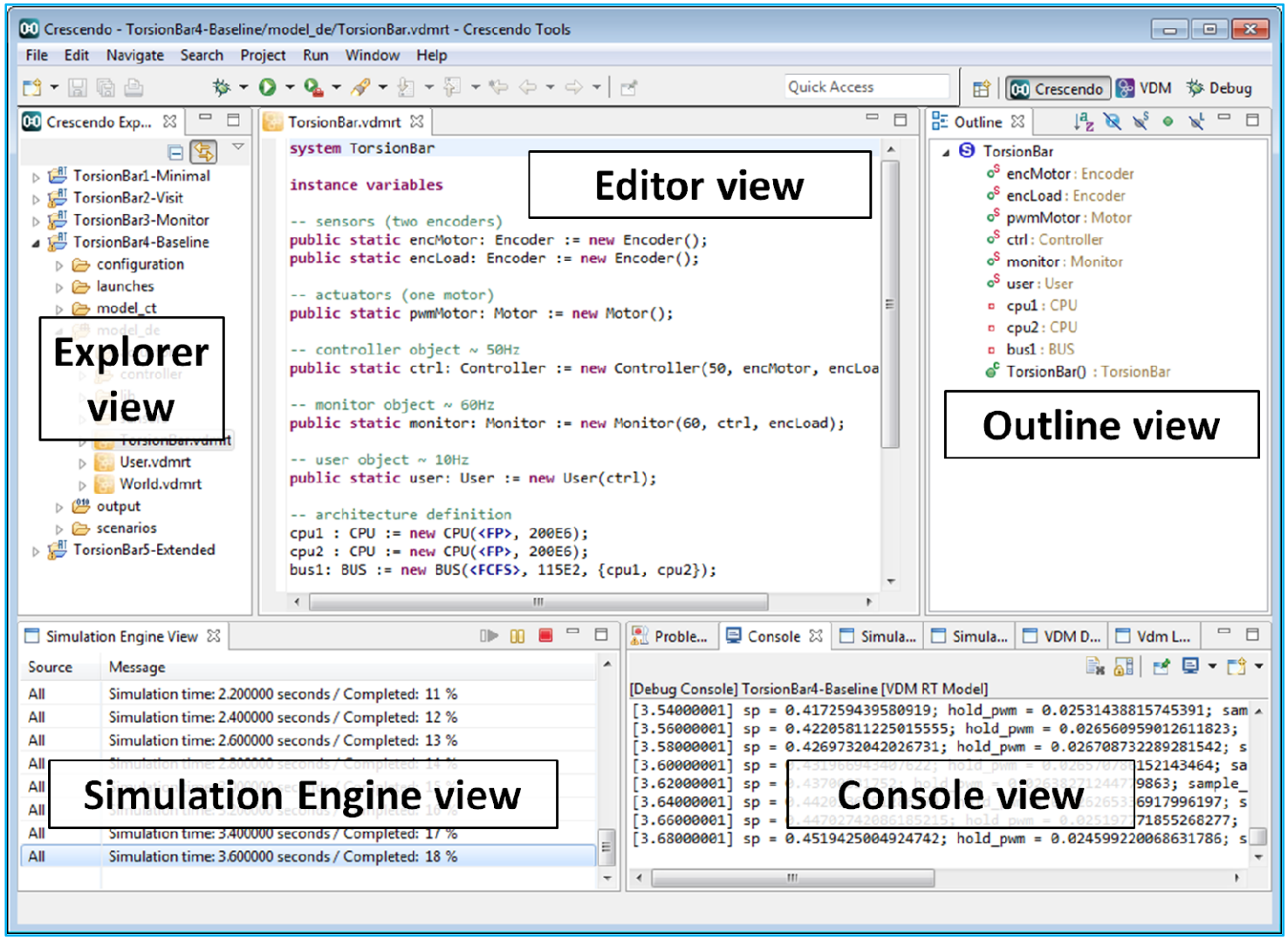
Co-simulationの意味論

- 各シミュレーターはローカルな状態と内部のシミュレーション時間を保持している.
- Co-simulationエンジンが下記を同期する.
  - 共有された変数，イベント，時間
- 共通の時間 t_n をco-simulationステップの開始とすると
- DEシミュレーターがステップの長さを決める（ロールバックを避けるため）.
- 時間 t_n においてDEシミュレーターは，
  - Controlled variablesの値を決める.
  - CTシミュレーターに対し経過時間を提案する（可能な場合）.
- Co-simulationエンジンが，CTシミュレーターに対しその経過時間だけ進むように指示する.

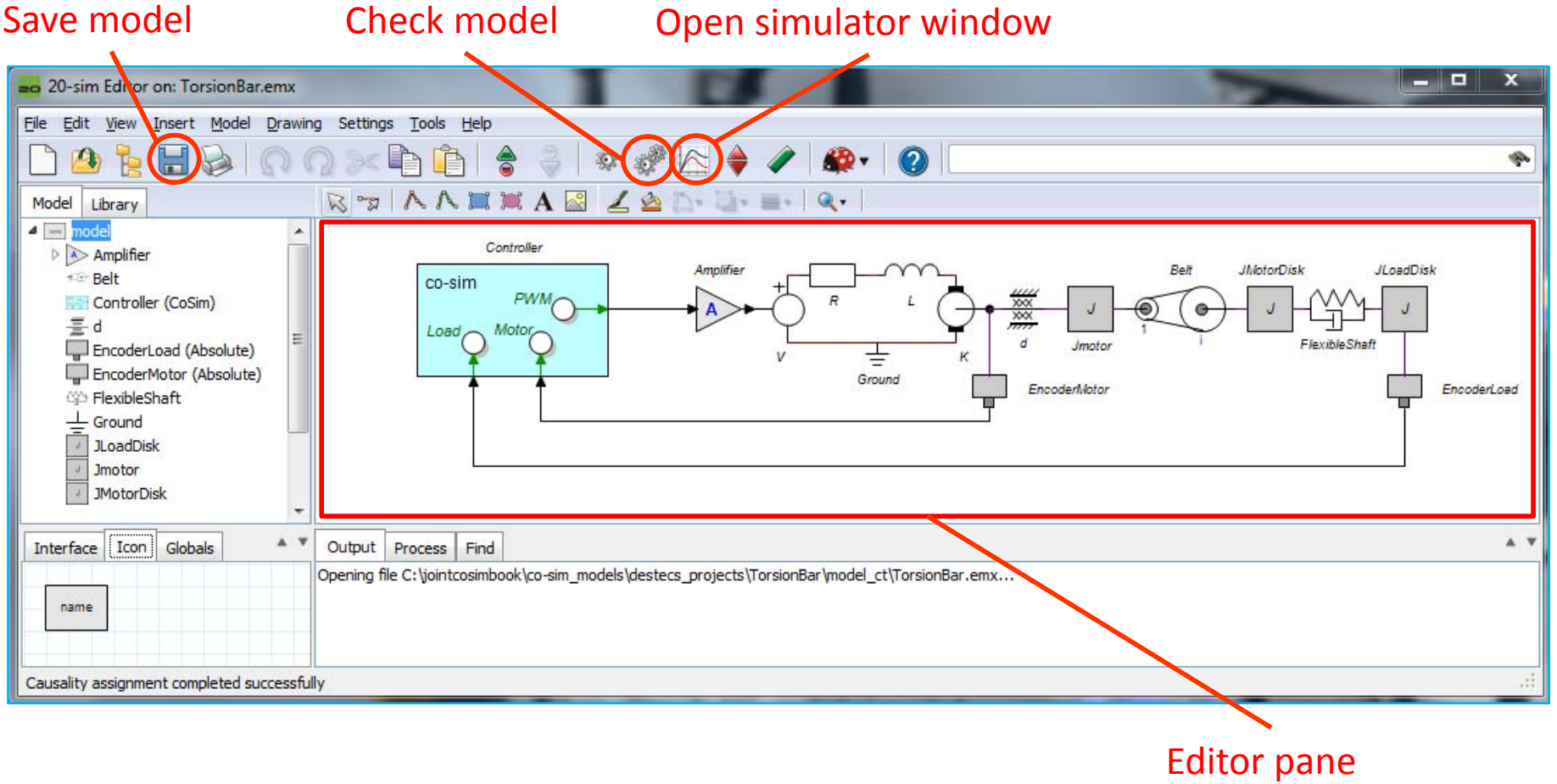
Co-simulationの意味論

- CTシミュレーターは時間を進める．もしも提案された経過時間に達する前にイベントが起きる場合，早く停止する．
- CTシミュレーターが停止をしたならば（内部時間 t_{n+1} とする），CTシミュレーション側で決められたmonitored variablesと実際の経過時間がDEシミュレーター側に返される．
- DEシミュレーションは，DE側とCT側が再度同じ時間で同期が取れるように進む
- *このサイクルを繰り返す．*

Crescendoのスクリーンショット



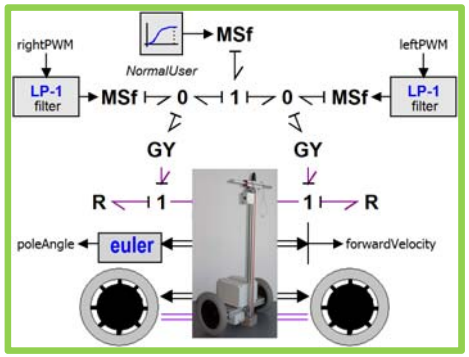
20-simのスクリーンショット



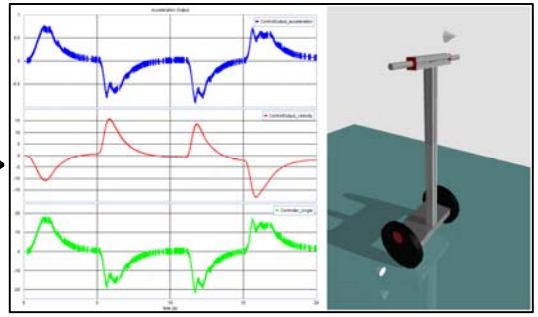
Example: Self-balancing Scooter

```
class Controller

instance variables
  -- sensors
  private angle: real;
  private velocity: real;
  -- actuators
  private acc_out: real;
  private vel_out: real;
  -- PID controllers
  private pid1: PID;
  private pid2: PID;

operations
  public Step : () ==> ()
  Step() == duration(20) (
    dcl err: real := velocity - angle;
    vel_out.Write(pid2.Out(err));
    acc_out.Write(pid1.Out(angle));
  );

  public GoSafe : () ==> ()
  GoSafe() == (
    vel_out.Write(0);
    acc_out.Write(0);
  );

thread
  periodic(1E6,0,0,0)(Step); -- 1kHz

end Controller
```

+

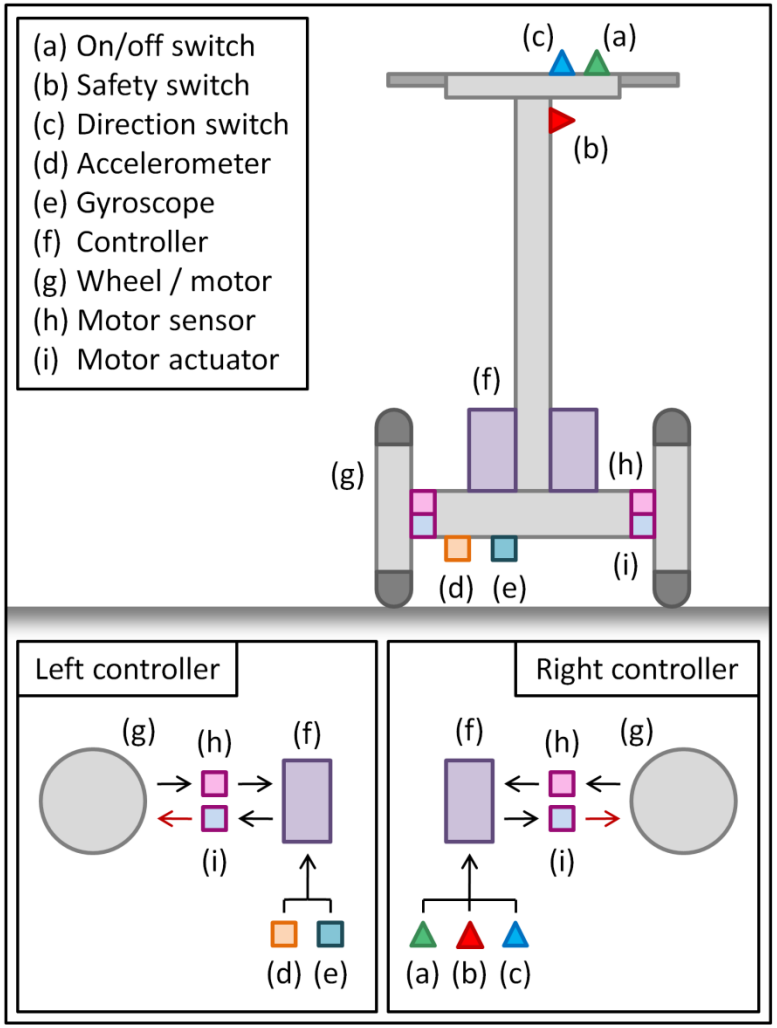


⇒

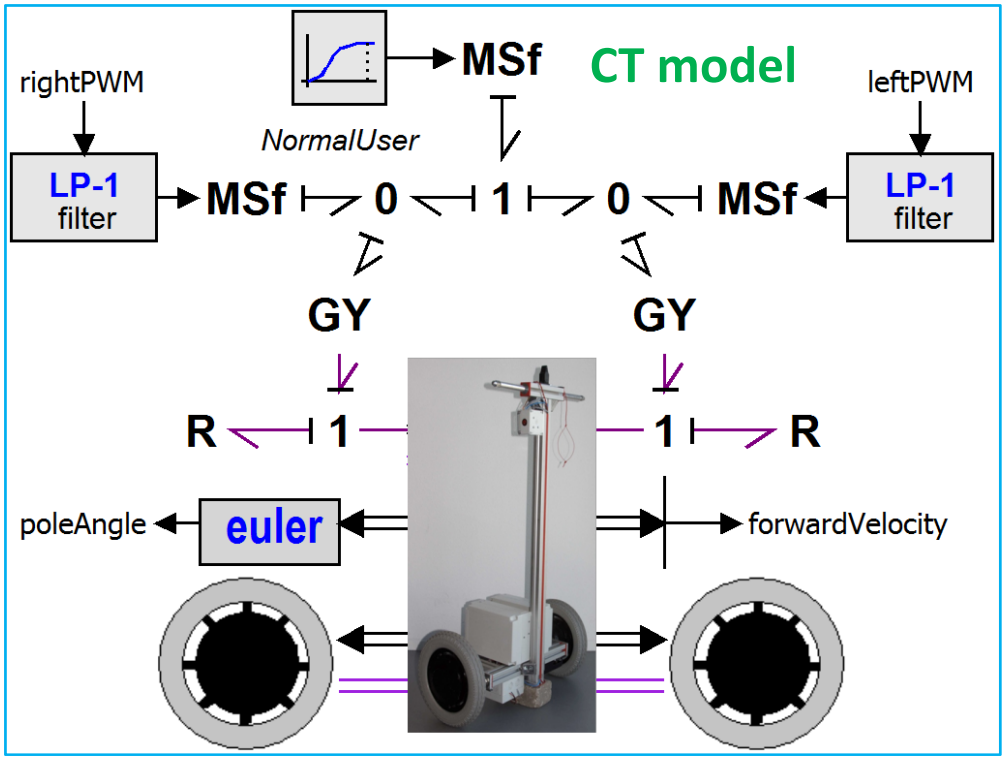


⇒

Example: Self-balancing Scooter



Example: Self-balancing Scooter



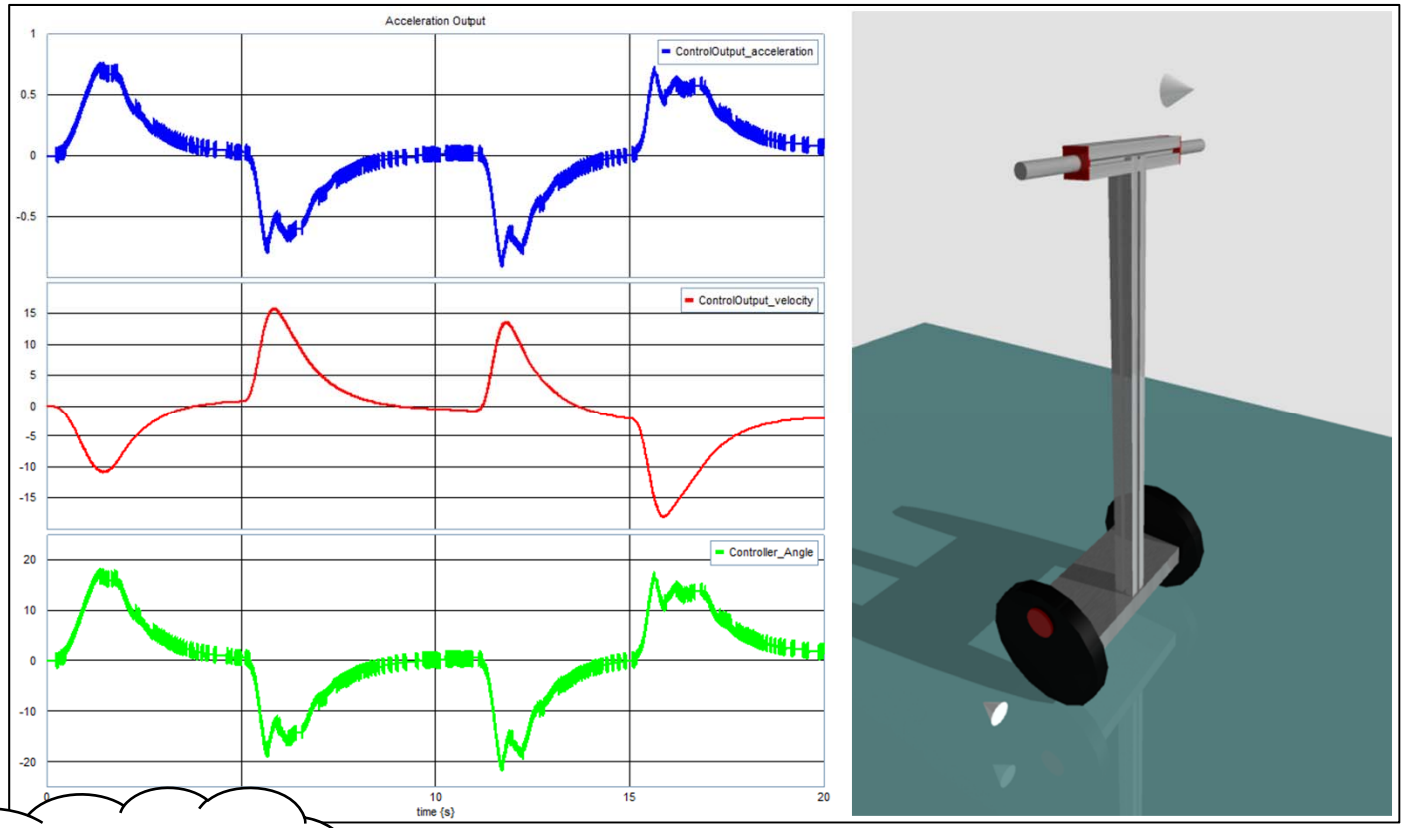
	Name	Type	Notes
controlled	leftPWM	real	range: [-1,1]
	rightPWM	real	range: [-1,1]
monitored	poleAngle	real	range: [0,2π]
	forwardVelocity	real	

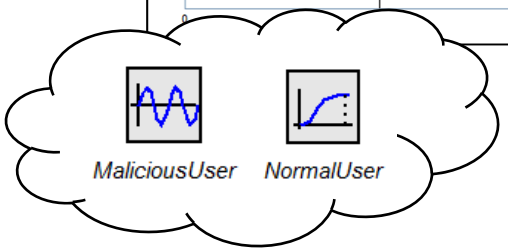
Contract

DE model

```
class Controller

instance variables
  -- sensors
  private angle: real;
  private velocity: real;
  -- actuators
  private acc_out: real;
  private vel_out: real;
  -- PID controllers
  private pid1: PID;
  private pid2: PID;

operations
  public Step : () ==> ()
  Step() == duration(20) (
    dcl err: real := velocity - angle;
    vel_out.Write(pid2.Out(err));
    acc_out.Write(pid1.Out(angle));
  );

  public GoSafe : () ==> ()
  GoSafe() == (
    vel_out.Write(0);
    acc_out.Write(0);
  );

thread
  periodic(1E6,0,0,0)(Step); -- 1kHz

end Controller
```

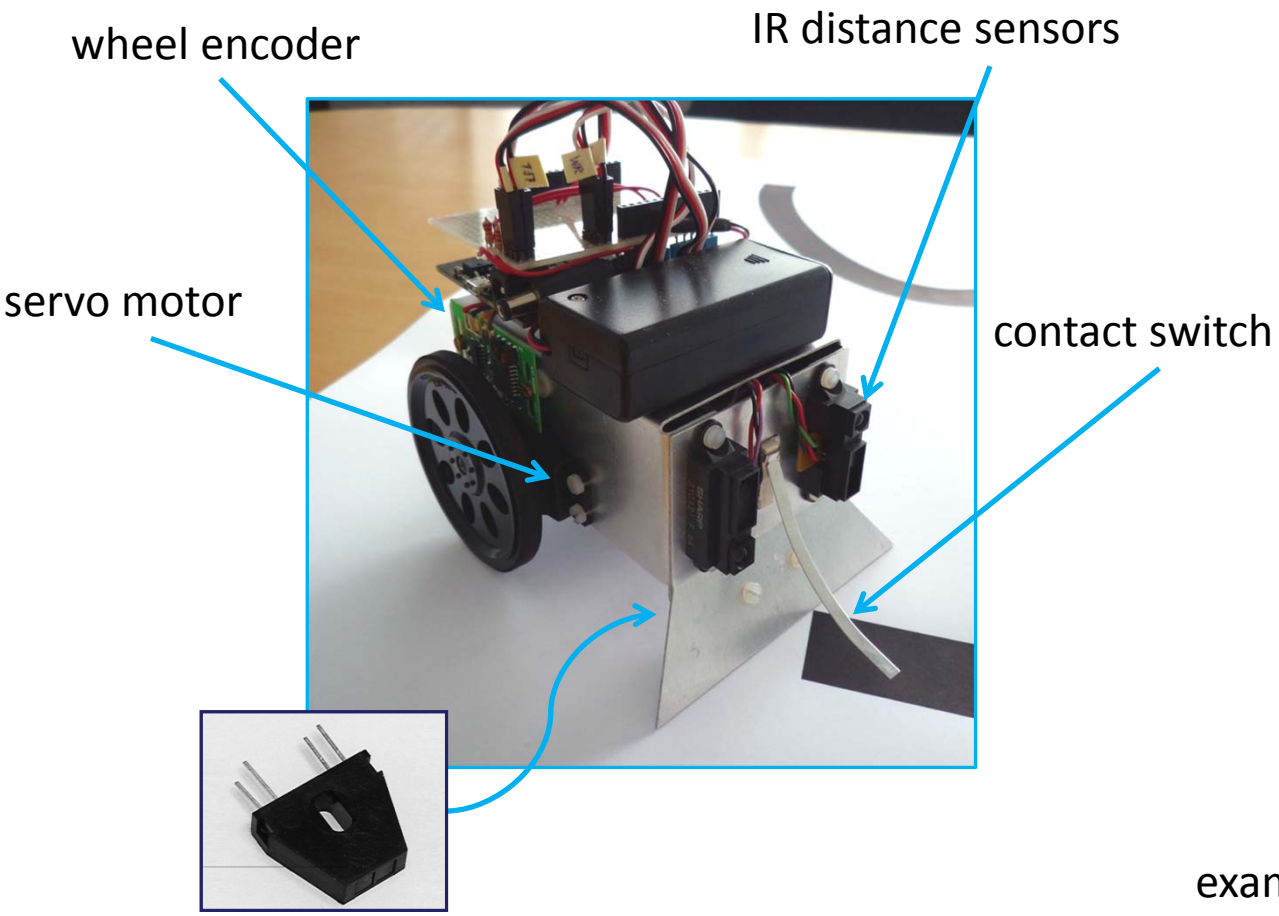
Example: Self-balancing Scooter

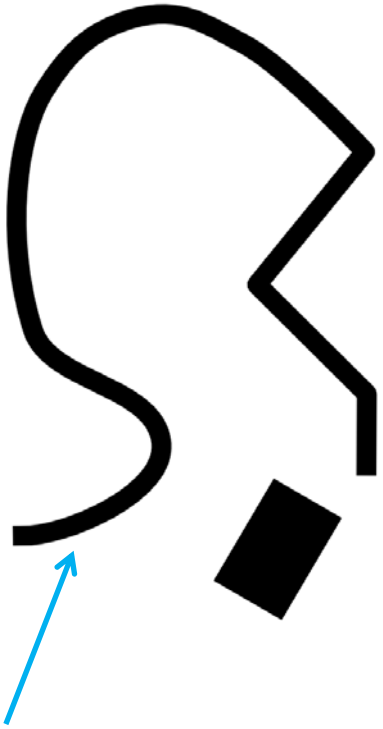


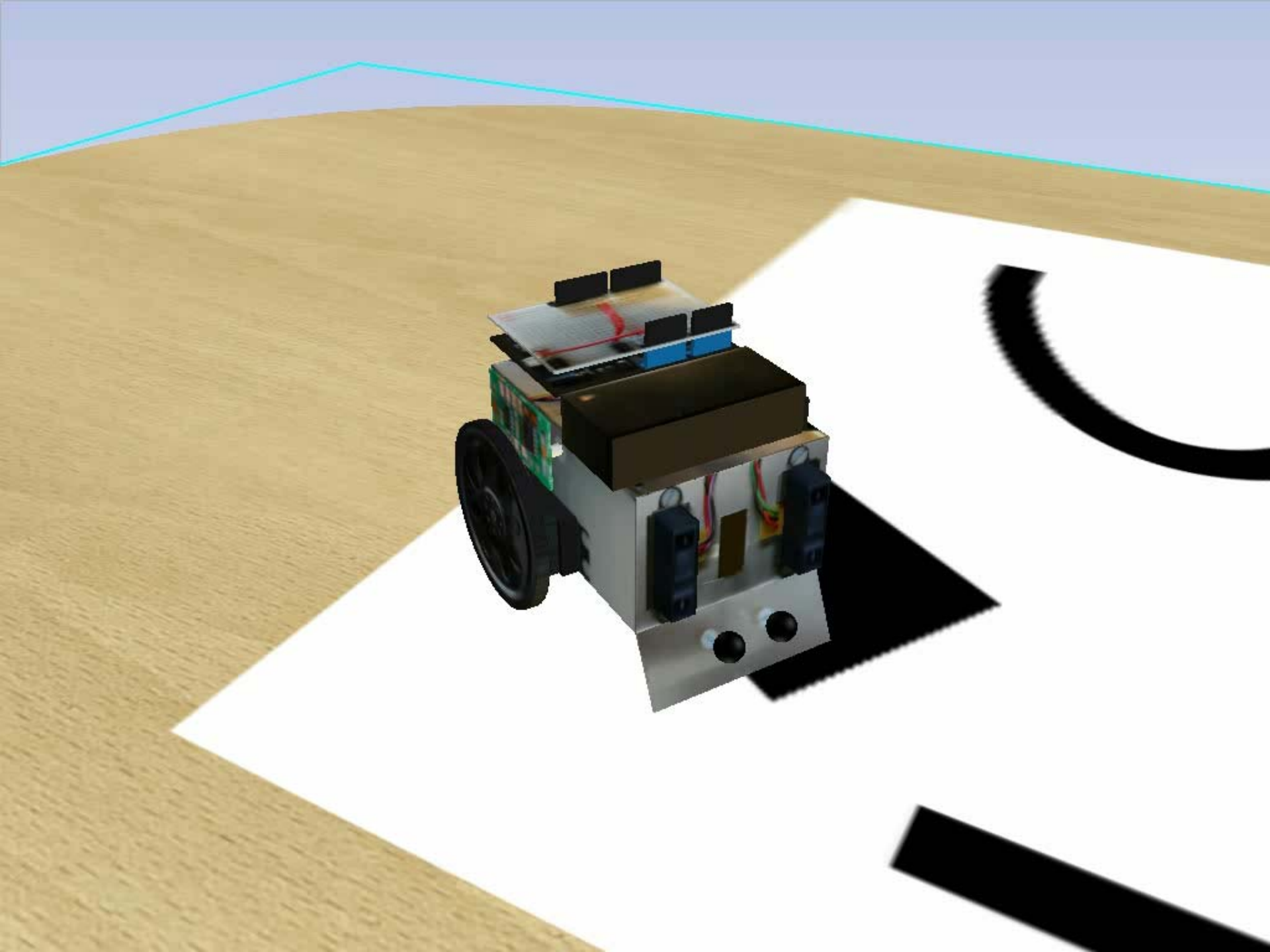


http://www.youtube.com/watch?v=pmLLGYn9Fo8

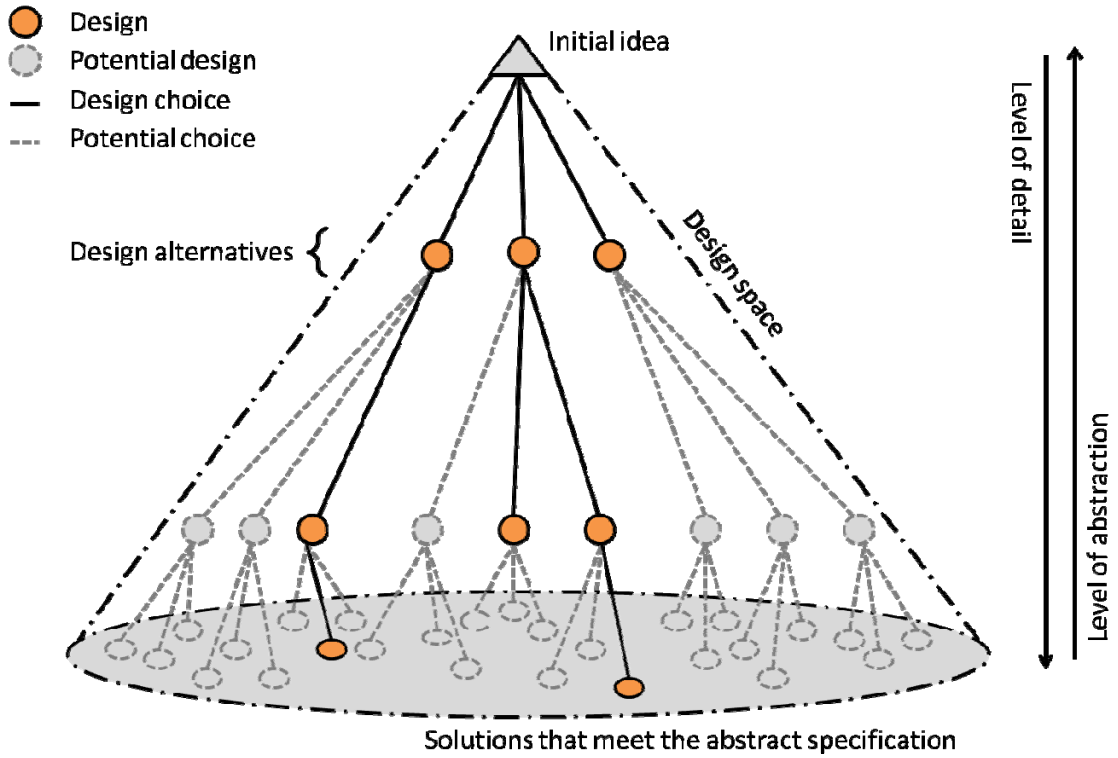
Example: Line-following Robot







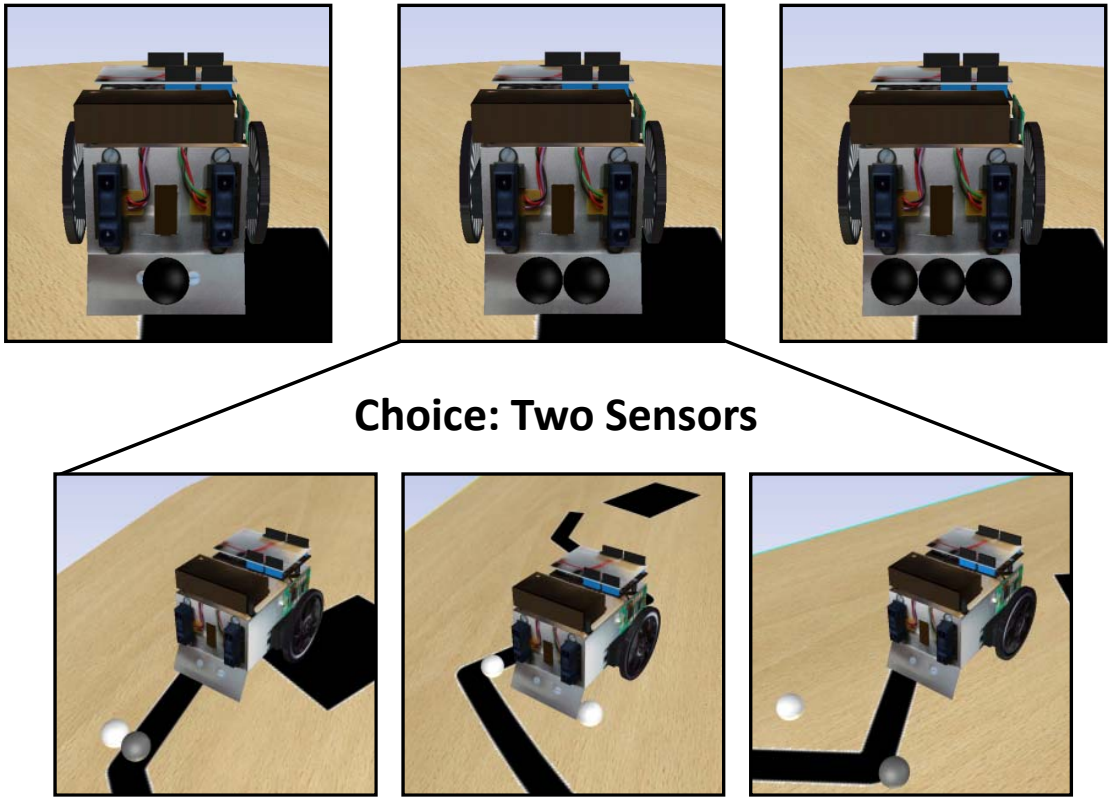
Design-Space Exploration (DSE)



- 設計の代替案から選択を行う（コストや性能などを踏まえ）
- 各地点で選ばれた代替により，以降のステップで可能となる設計の範囲が決まる

Line-following Robot DSE

- 設計の選択により設計空間が制限される
- 探索は決定を行っていくことである

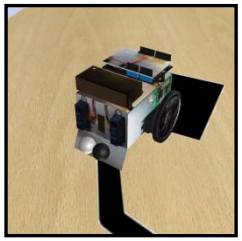


Line-following Robot DSE

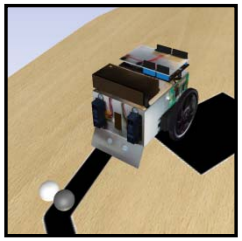
縦のセンサー

側面のセンサー

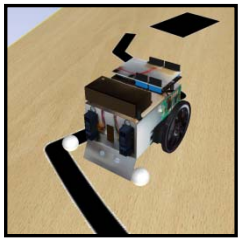
		Longitudinal sensor offset		
		0.01m	**0.07m**	**0.13m**
Lateral sensor offset	**0.01m**	(a)	(b)	(c)
	0.03m	(d)	(e)	(f)
	0.05m	(g)	(h)	(i)



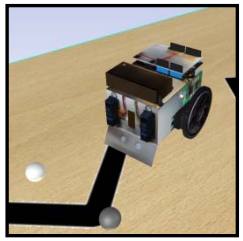
(b)



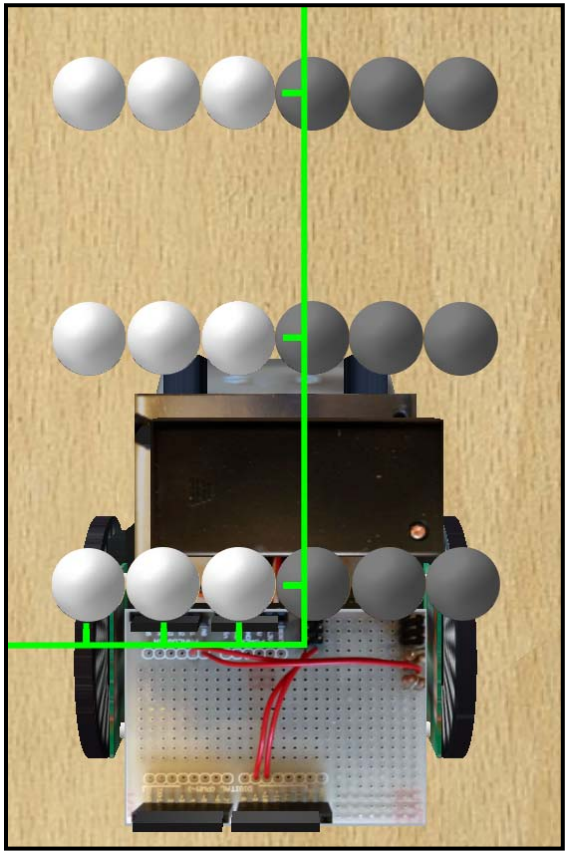
(c)



(h)



(i)

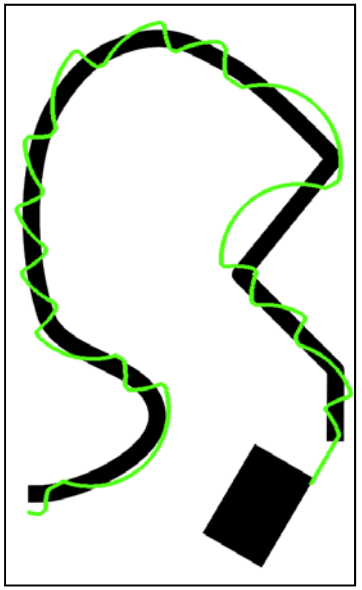


- 2つの設計パラメーターによる9個の選択肢・シミュレーション

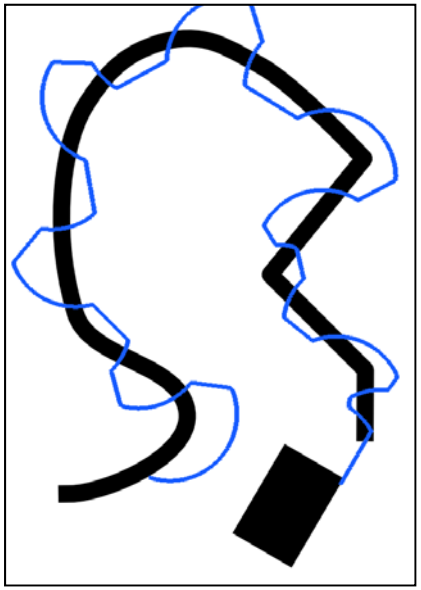
Line-following Robot DSE

- 結果はグラフィカルにまたは数値で表示
- 設計はランク付け関数により評価できる

		Metric*				
Rank	Design	A	B	C	D	Mean Rank
1	**(b)**	**1**	**5**	**1**	**2**	**2.2**
2	(f)	7	2	4	1	3.5
3	(a)	2	8	2	4	4.0
4	(e)	3	6	3	5	4.2
5	(i)	9	1	5	3	4.5
6	(c)	5	3	6	8	5.5
7	(d)	6	4	7	7	6.0
8	(h)	4	7	8	9	7.0
9	(j)	8	9	9	6	8.0



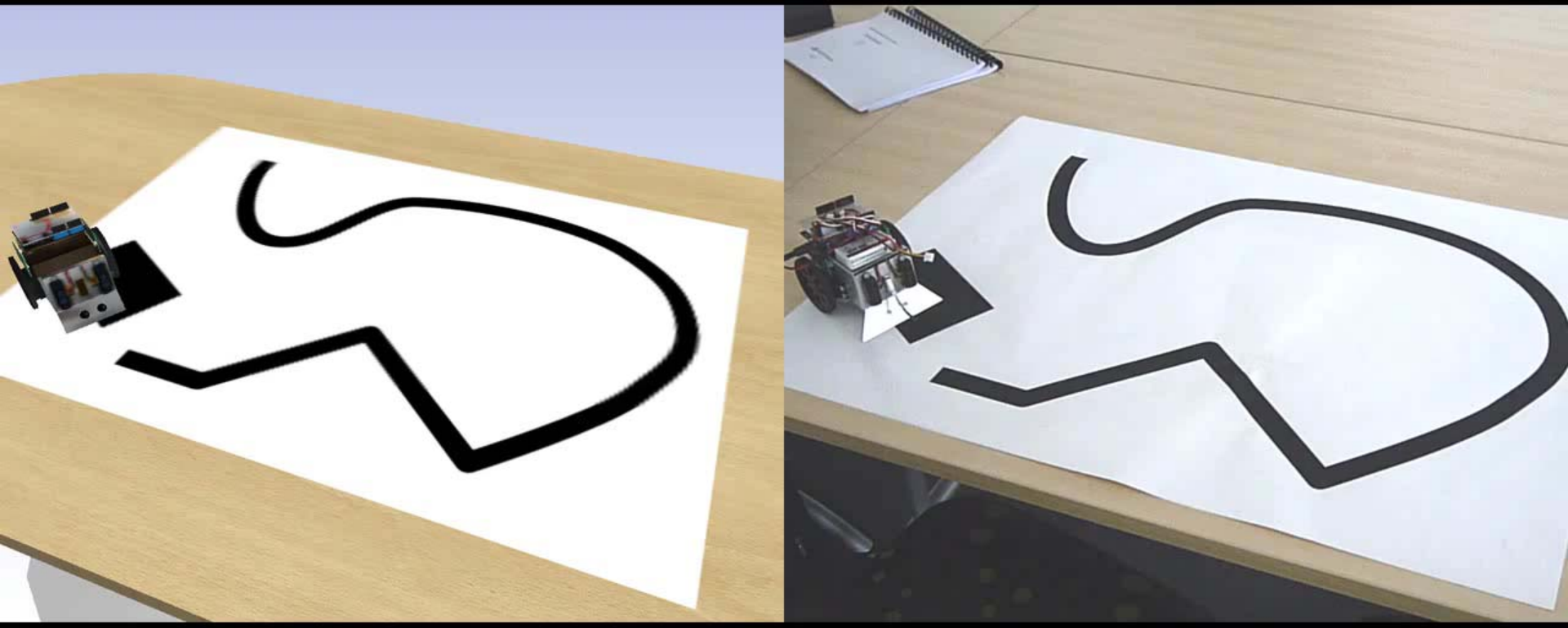
(b)



(h)

*** A = distance, B = energy, C = deviation area, D = maximum deviation**

Line-following Robot

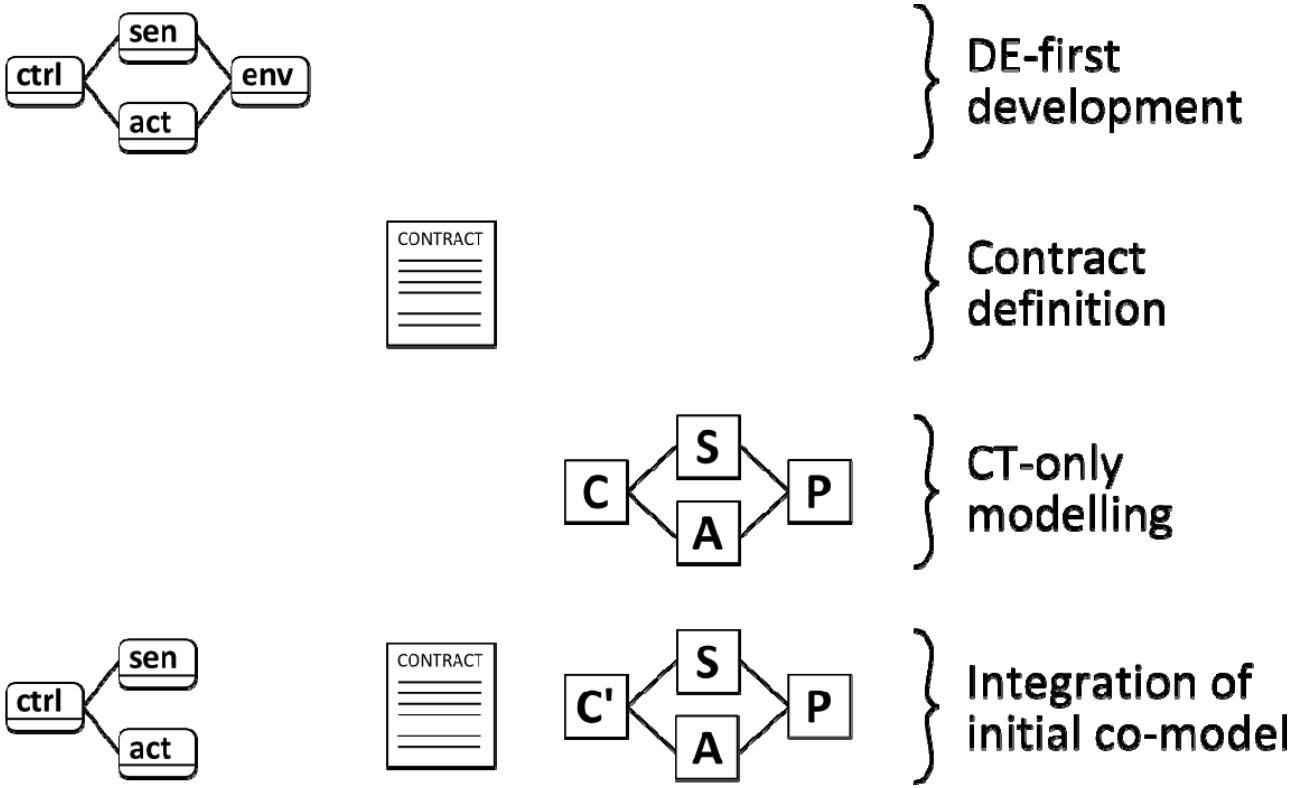


初期のCo-modelsへの道筋

- **DE-first**
  - 離散イベントの形式化に基づき初期モデルを構築し，CTモデルを後で追加する．DEコントローラーにまず集中する．
- **CT-first**
  - CTツールを用いて初期モデルを構築し，DEモデルを後に導入してco-modelを形成する．動特性のモデリングに集中する．
- **Contract-first**
  - ガイドとしての契約を定義する．DEモデル，CTモデルを別々に，並行に開発する（上記双方を行う）．対となるモデルが適切にできあがるまで待たずに，構成モデルの早期のテストが可能となる．構成モデルを統合してco-modelを得る．

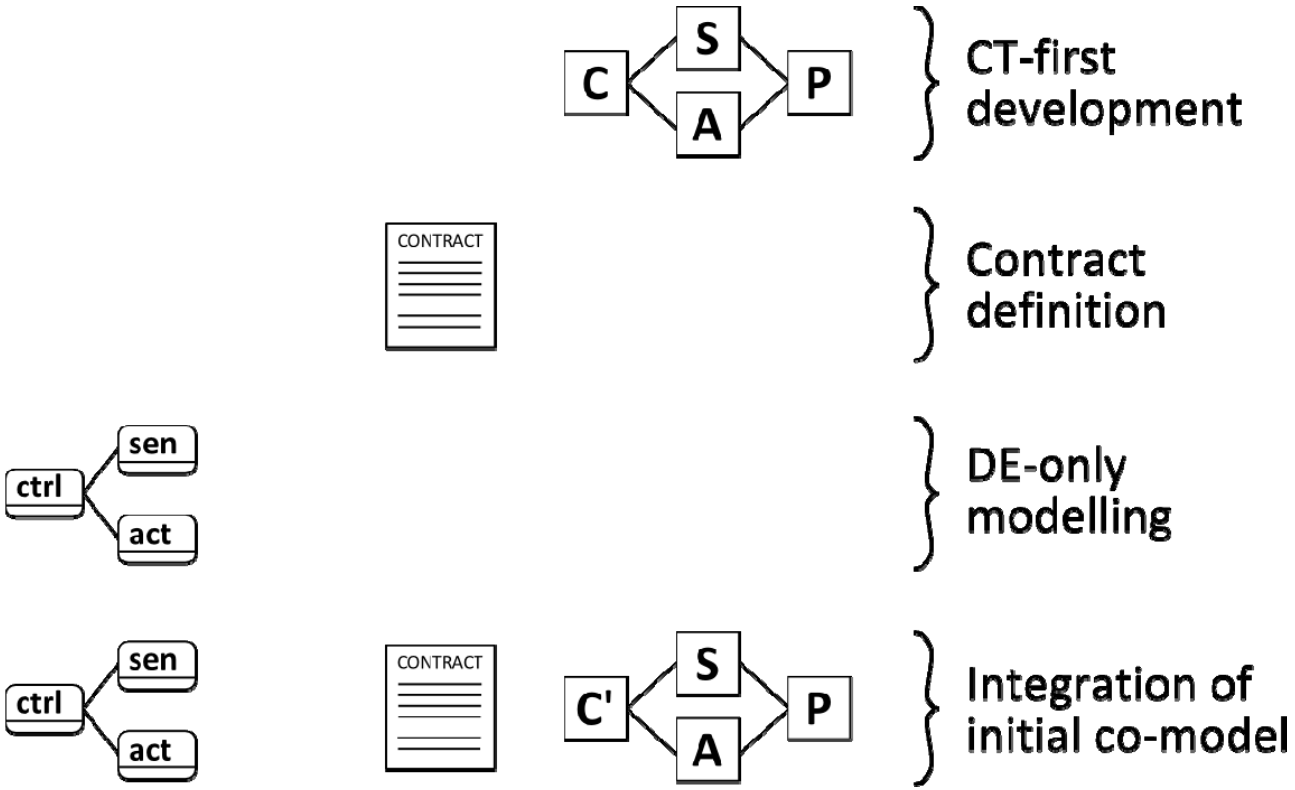
DE-first

- 離散イベントの形式化に基づき初期モデルを構築
- CTモデルを後で追加



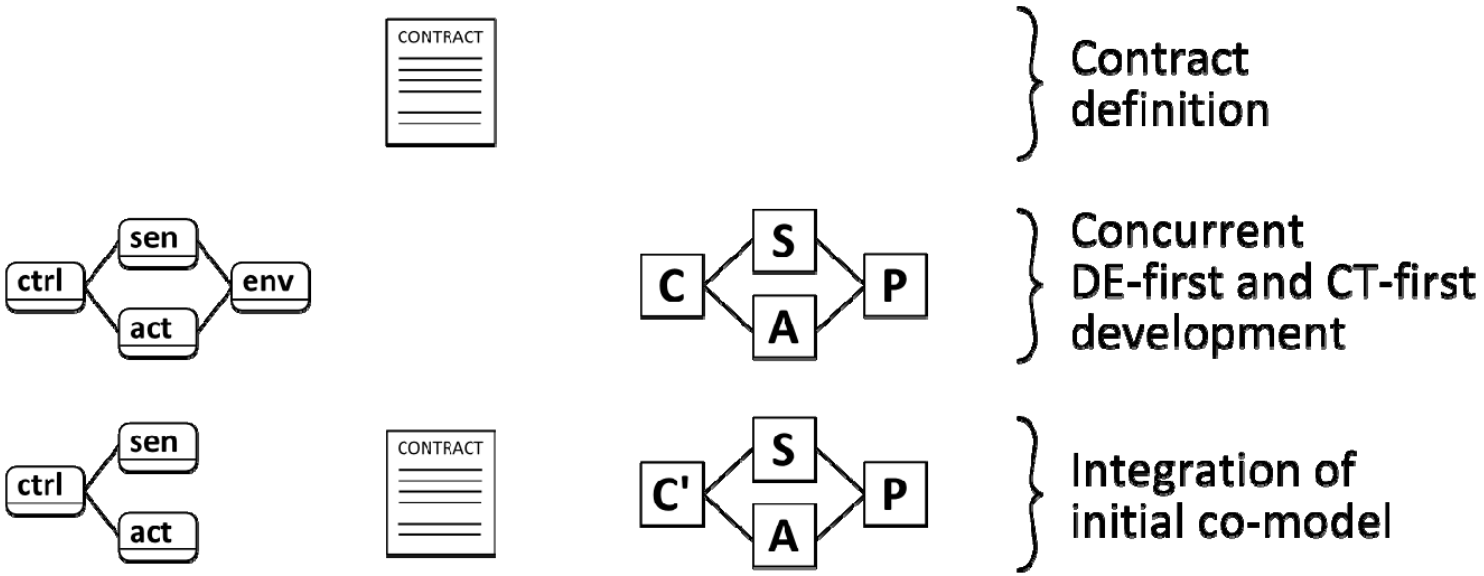
CT-first

- CTツールを用いて初期モデルを構築
- DEモデルを後に導入してco-modelを形成



Contract-first

- ガイドとしての契約を定義
  - 構成モデルの早期のテスト
- DEモデル，CTモデルを別々に，並行に開発
- 構成モデルをco-modelへと統合



道筋の選択

	利点	欠点	用いる状況
DE-first	複雑な制御の振る舞いを早期に追求できる	動特性が過度に単純化される．ループコントローラーのチューニングができない．環境モデルの複雑さが速く増大．	複雑なDE制御を優先したい．レガシーなDEモデルがすでに存在する．モデラーの経験が主にDEにある．
CT-first	フィジビリティーの調査ができる．動特性を早期に追究できる，ループコントローラーのチューニングができる．	複雑なDE制御を早期に追求できない．	制御のフィジビリティーが不明．動特性を優先したい．レガシーなCTモデルあるいはループコントローラーが存在する．モデラーの経験が主にCTにある．
Contract-first	早期にco-modelが得られる．構成モデルが互いに依存せずテストできる．	契約を早期に確立しなければならない．構成モデルのテスト環境に追加の労力が必要となる．	2つのレガシーモデルの統合が必要である．レガシーモデルが存在しない．双方の流域のモデラーが存在する．
Other	既存のプラクティスに応じてあたらいいアプローチがよりうまく可能性がある．	我々の既存のガイドラインからの経験は限られている	標準のアプローチは開発の状況に合わない．他の形式化に基づくレガシーなモデルや経験がある．

用語のまとめ (1)

- **model**
  - 興味の対象となるシステムや部品の，多かれ少なかれ抽象化された表現
- **modeling**
  - モデルを構築する活動
- **simulation**
  - モデルのシンボリックな実行
- **continuous-time simulation**
  - 時間の経過に応じてシステムの状態が連続的に変化するシミュレーションの形式.
- **discrete-event simulation**
  - システムの状態が変化する時間ポイントだけが表現されたシミュレーションの形式

用語のまとめ (2)

- **co-model**
  - 2つの構成モデル（DE submodelとCT submodel）およびそれらの間の相互作用に関する契約からなるモデル
- **contract**
  - 共有される設計パラメーター，変数，イベントの形式で与えられる，構成モデル間の相互作用に関する記述
- **co-simulation**
  - co-modelのシミュレーションthe simulation of a co-model.
- **design space exploration (DSE)**
  - co-modelを構築し，，co-simulationを実施し結果を評価することにより，次の反復におけるco-modelを選択する（反復的な）

まとめ

- 組み込みシステムの設計
  - 協調的な開発への要求
  - 異なる領域からのモデルの分析
  - 多様な文化，抽象化，形式化
- Crescendoにおけるソリューションは **co-simulation**
  - コントローラーのDEモデルと，コントロールされるプラントのCTモデルとを組み合わせる
  - すでにある知識，スキルを利用可能にする
  - 領域間のコミュニケーションを促進する