

Assured and Correct Dynamic Update of Controllers

Kenji Tei

Associate Professor

National Institute of Informatics, Japan

tei@nii.ac.jp

<http://researchmap.jp/teikenji/?lang=english>


In collaboration with Leandro Nahabedian, Victor Braberman, Nicolas D'Ippolito, Shinichi Honiden, Jeff Kramer, and Sebastian Uchitel.

This work was presented at SEAMS 2016 and selected as the best paper

National Institute of Informatics
情報学研究所

Kenji Tei

- Associate Professor, National Institute of Informatics, Japan
- Research interests:
 - Software architecture, MDD
 - **Self-adaptive system, Models@run.time**
 - IoT / CPS / Smart city
- Projects
 - **TopSE**: Education program for software engineers in industry
 - <http://www.topse.jp/en/>
 - **ClouT/BigClouT**: EU-JP joint project about smart cities
 - <http://bigclout.eu>



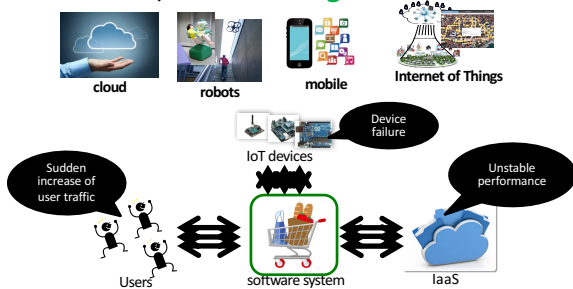
Summary

- Context
 - Environment will change at runtime
 - How do we ensure correctness of software?
=> Use models at runtime for self-adaptation!
- Dynamic update of controller
 - Software (controller) will be changed at runtime in response to changes in the environment
 - How do we ensure that update of controller is correct?
=> Synthesize updating controller!

Context

How does the system adapt to changes?

Modern systems face **changes** at runtime



How do we ensure correctness of software system?

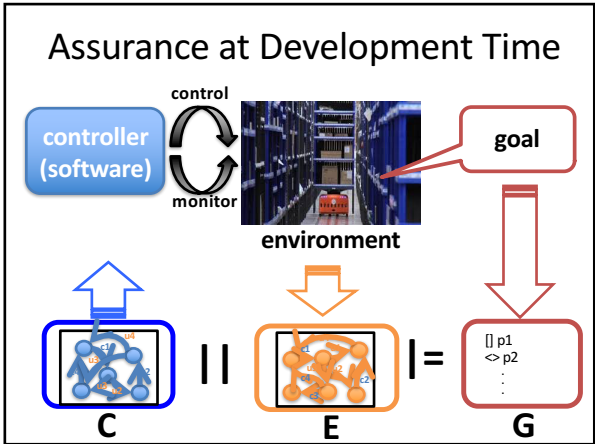
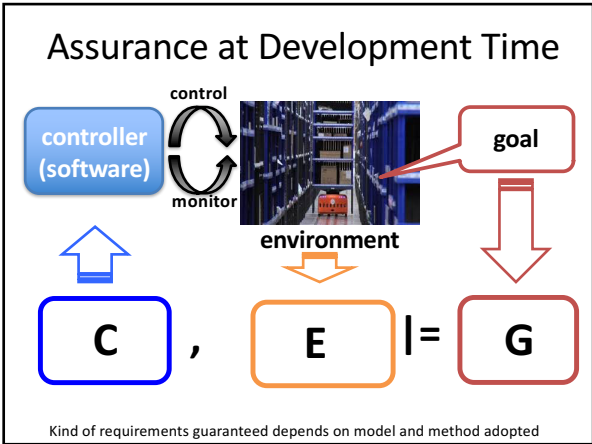
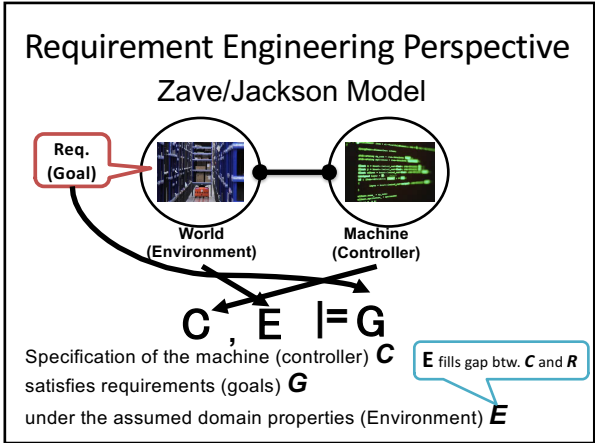
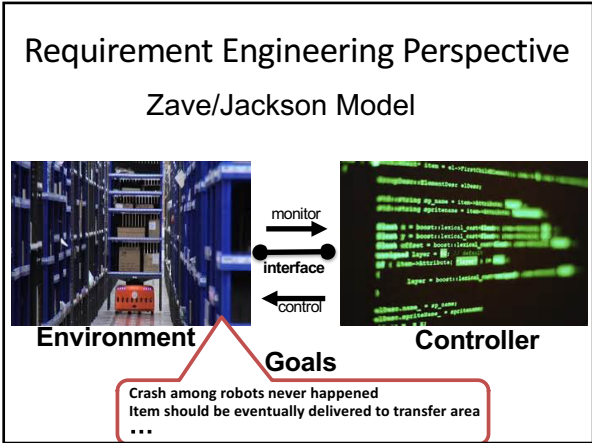
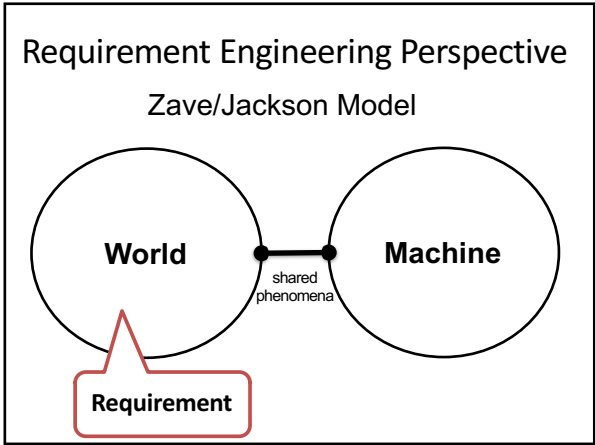
Self-adaptive Systems

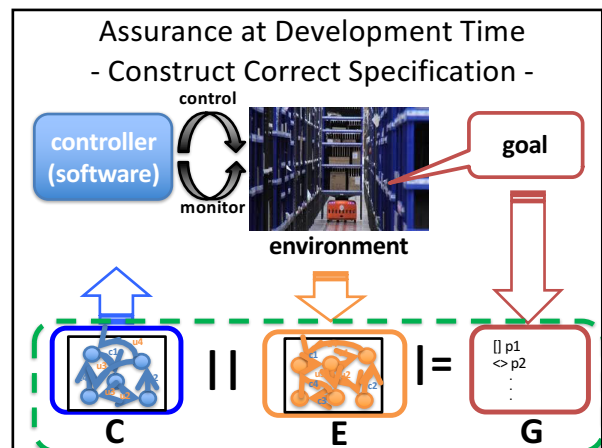
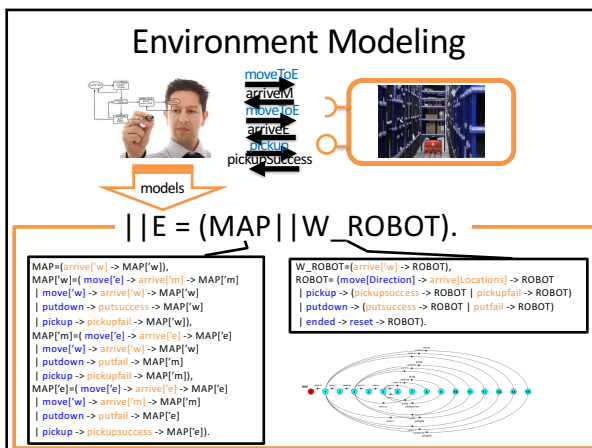
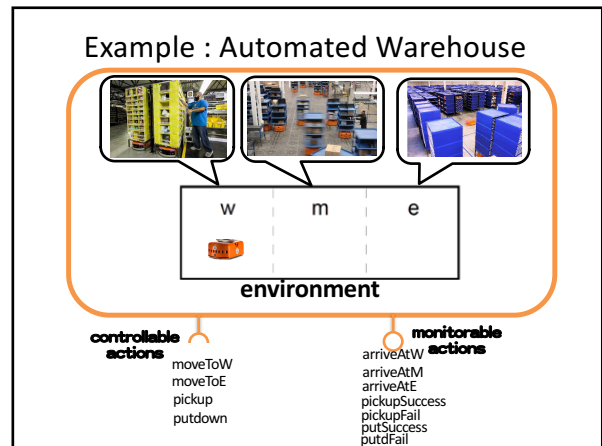
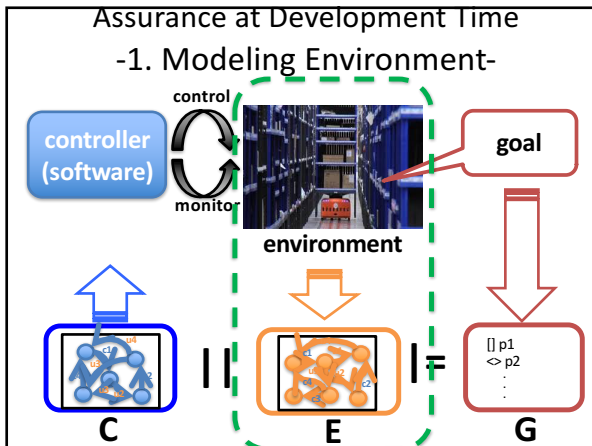
Systems that are able to **modify their behavior and/or structure** in response to **their perception of the environment and the system itself**, and **their requirements**

R.Lemos, et al., Software Engineering for Self-Adaptive Systems: A Second Research Roadmap, SEAMS2011.

Why self-adaptation is needed?

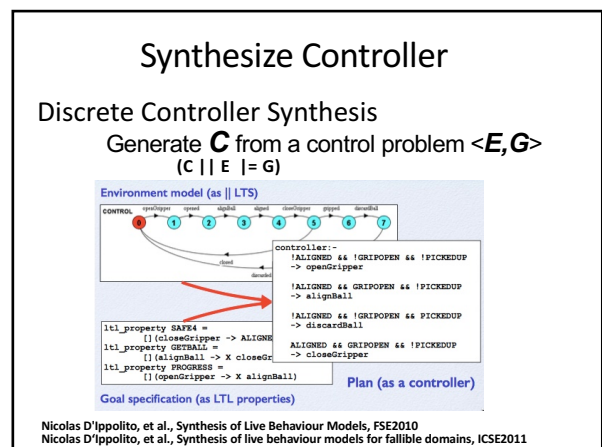
How is its correctness guaranteed?



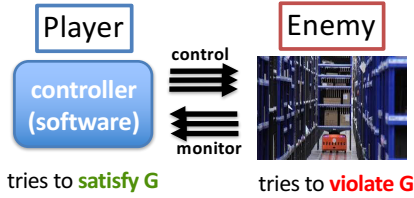


Construct Correct Controller Specification

- by hand
 - Developer specifies controller,
 - checks correctness of it by model checking
- by automatic generation
 - Tool generates correct specification for the formally modeled environment and goals



- Discrete Controller Synthesis-
Synthesis as 2 Player Game

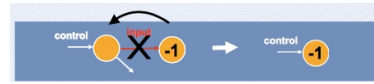


Is there a **winning strategy** for the player?

- Discrete Controller Synthesis-
Synthesis as 2 Player Game

Compute winning states

- backward propagation error states for input



- ... for control



- Discrete Controller Synthesis-
Synthesis as 2 Player Game

Extract winning strategy

Reactive plan computed from set of control state

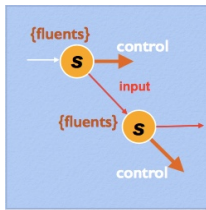
```

controller:-
  !ALIGNED && !GRIPOPEN && !PICKEDUP
  -> openGripper

  !ALIGNED && GRIPOPEN && !PICKEDUP
  -> alignBall

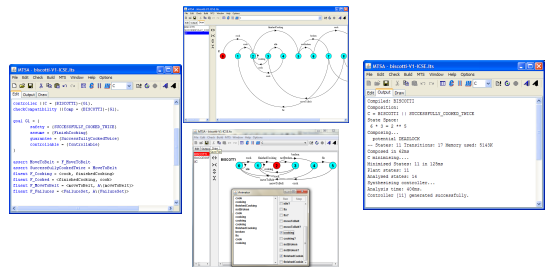
  !ALIGNED && !GRIPOPEN && PICKEDUP
  -> discardBall

  ALIGNED && GRIPOPEN && !PICKEDUP
  -> closeGripper
  
```



- Discrete Controller Synthesis-
Tool Support

- MTSA (Modal Transition System Analyzer)



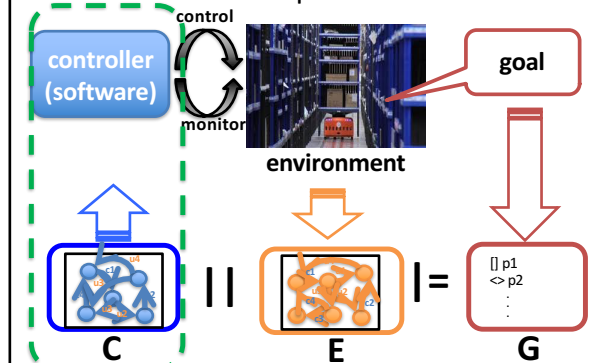
<http://mtsa.dc.uba.ar>

- Discrete Controller Synthesis-
Theoretical Complexity

Linear Temporal Logic	2-EXPTIME
Generalized Reactivity (k) $\bigwedge_{i=1}^k (\bigwedge_{j=1}^m \square p_i \vee \bigwedge_{j=1}^m \square q_j)'$	EXPTIME
Generalized Reactivity (1) $(\bigwedge_{i=1}^m \square p_i \vee \bigwedge_{j=1}^m \square q_j)'$	Polynomial
Persistence $\square \square q$	Polynomial
Response $\square \square p$	Polynomial
Obligation $\bigwedge_{i=1}^k (\square p_i \vee \square q_i)'$	Linear
Reachability $\diamond q$	Linear
Safety $\square p$	Linear

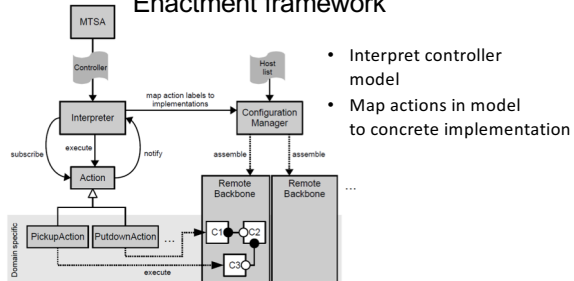
Assurance at Development Time

-3. Develop Software-



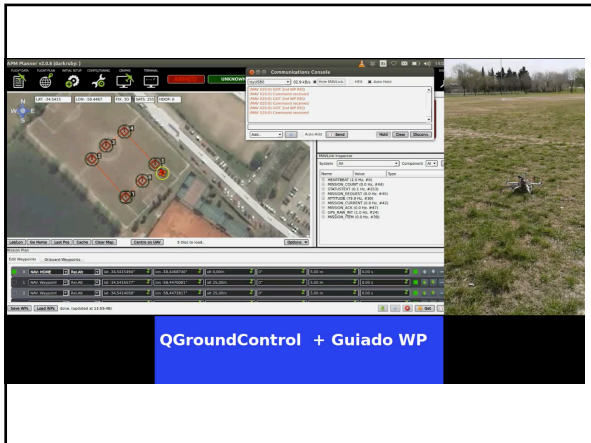
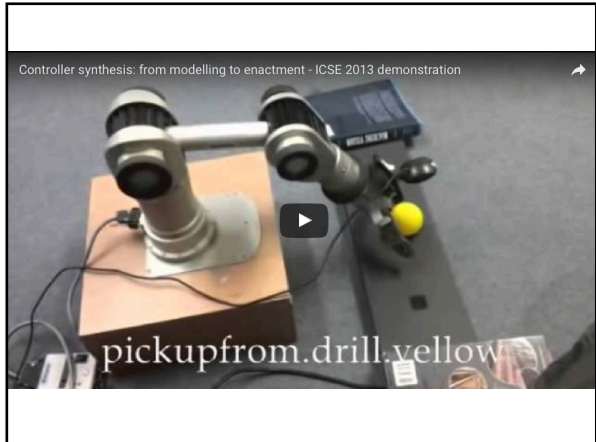
Enact Model

Enactment framework

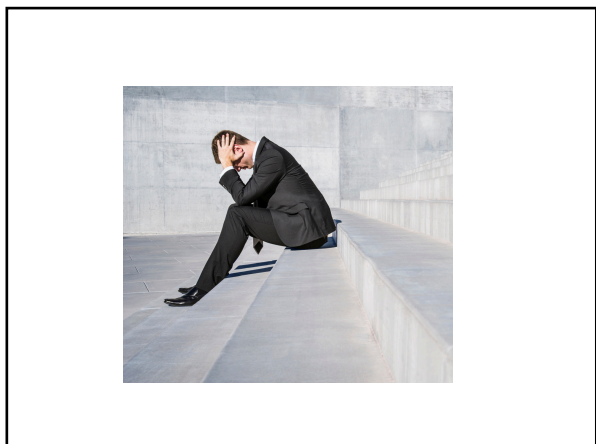
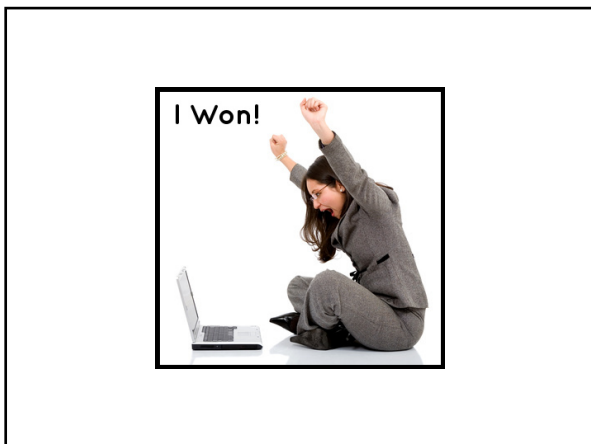
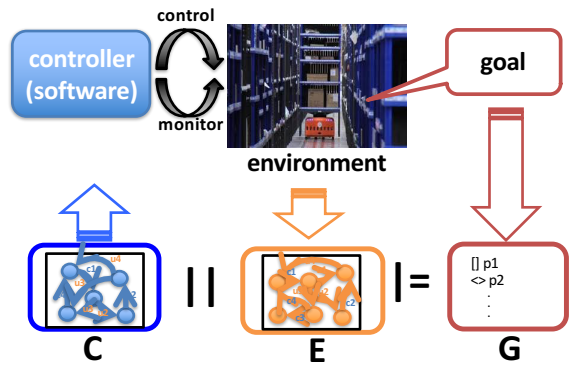


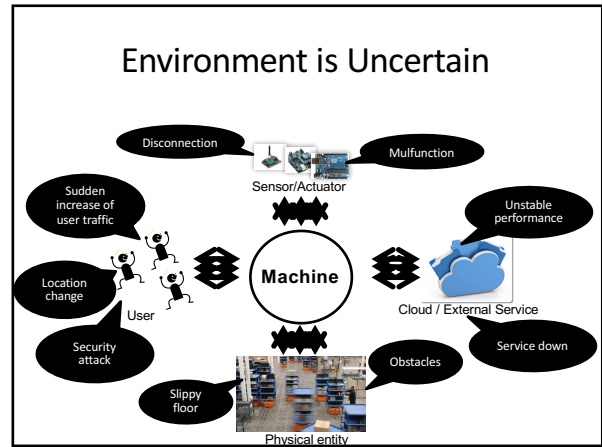
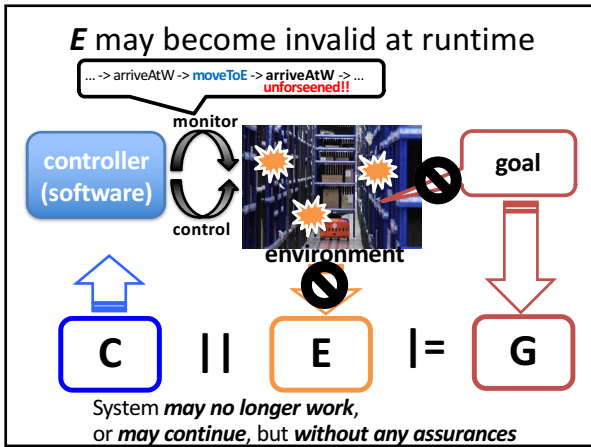
- Interpret controller model
- Map actions in model to concrete implementation

V.Braberman et al., Controller synthesis: From modelling to enactment, ICSE 2013



Assurance at development time





Challenge to Uncertainty

by David Garlan

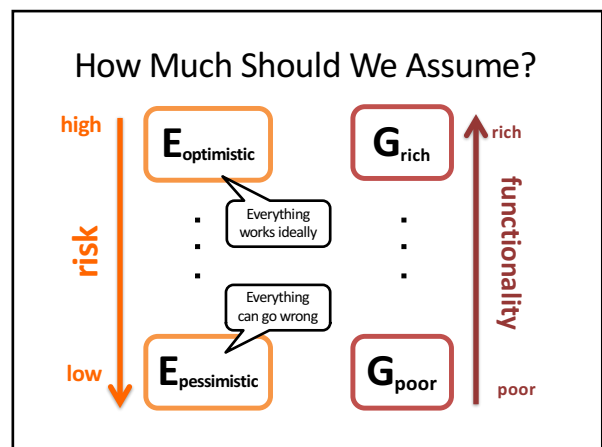
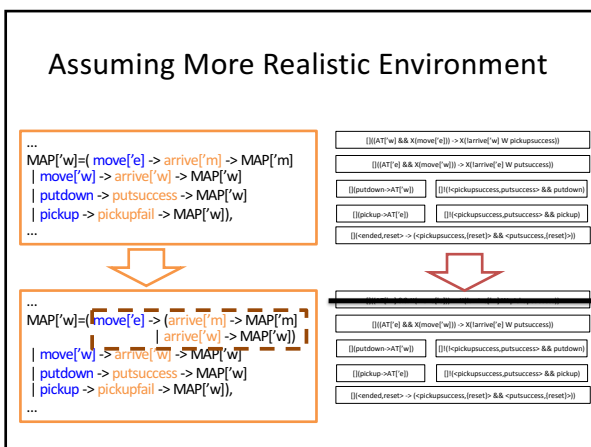
Software Engineering is founded on a computational **myth** that no longer fully serves its purpose: that the **computational environment is predictable and in principle fully specifiable**, and that **the systems that compute in those environments can in principle be engineered so that they are trouble-free**

D. Garlan, Software Engineering in an Uncertain World, 2010

How to Address Uncertainty

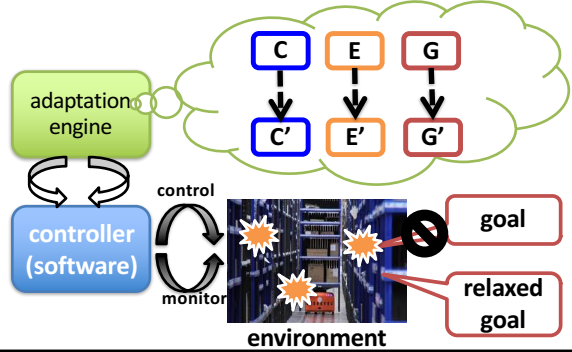
- **Do nothing**
- **Acknowledge it**
- **Delay key decisions** until more information is available

B. Cheng, A Search-based Approach to Exploring Uncertainty for Self-Adaptive Systems, 2016

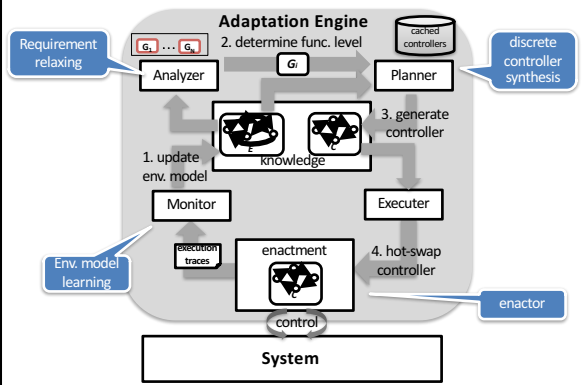


Use models at runtime!

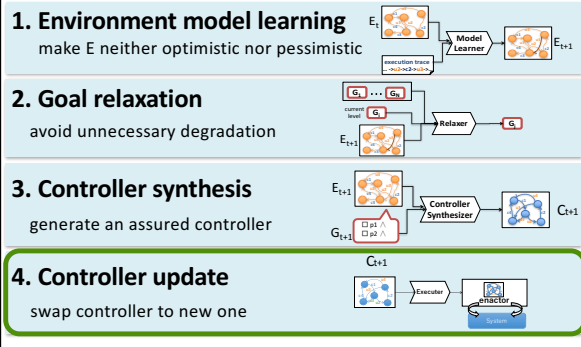
Graceful Degradation by Self-adaptation with Models



Self-adaptation by Models@run.time



Key Techniques



How do we ensure correct update of controller?

Assured and Correct Dynamic Update of Controllers

Kenji Tei

Associate Professor

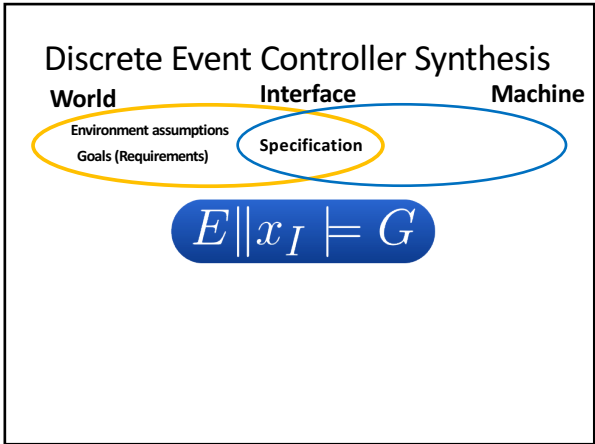
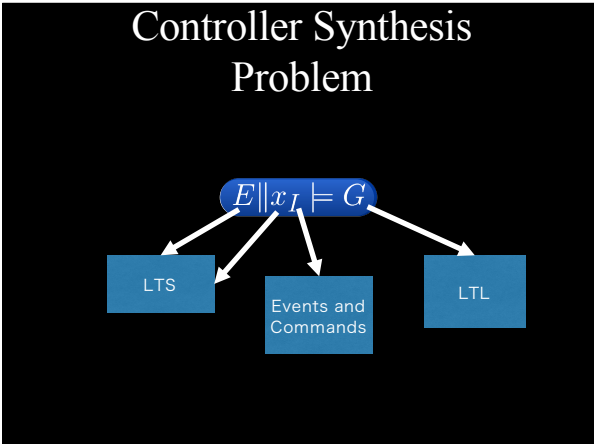
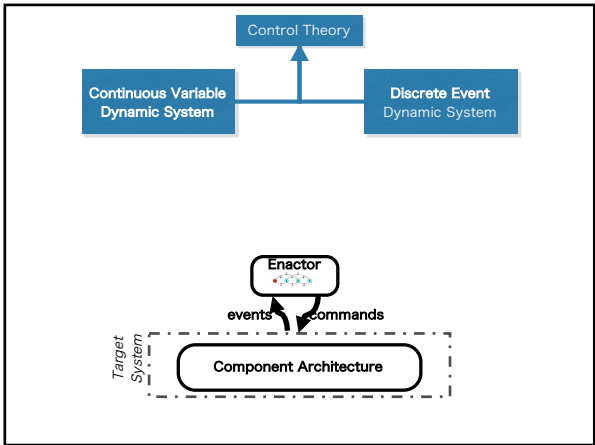
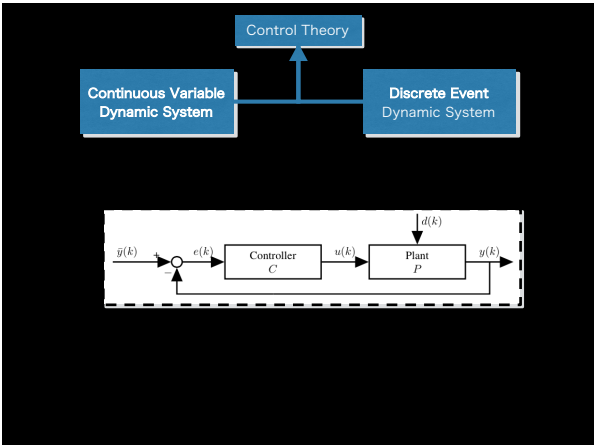
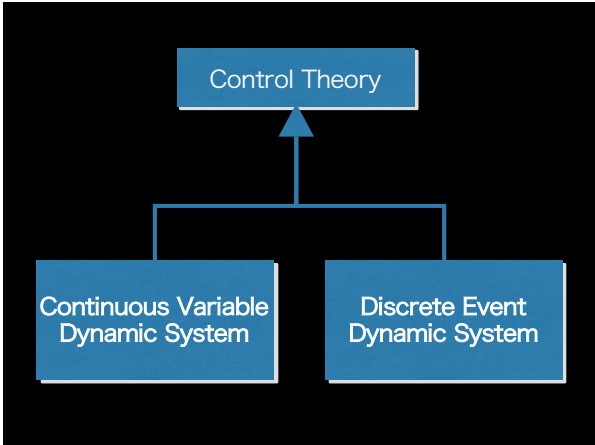
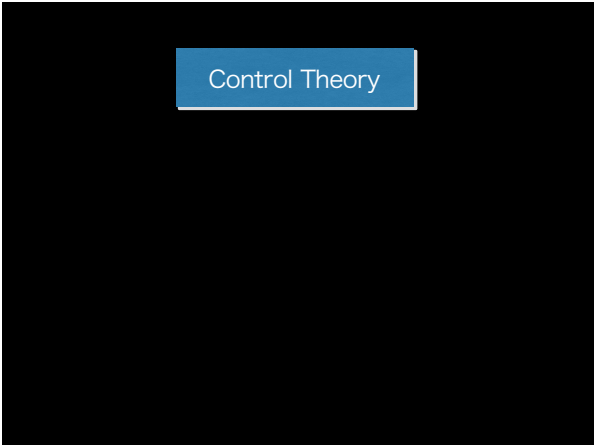
National Institute of Informatics, Japan

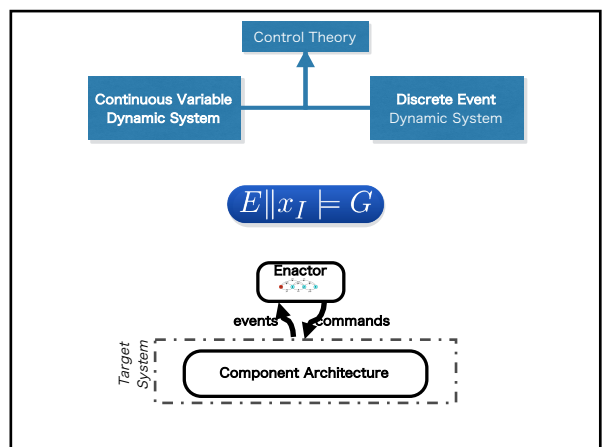
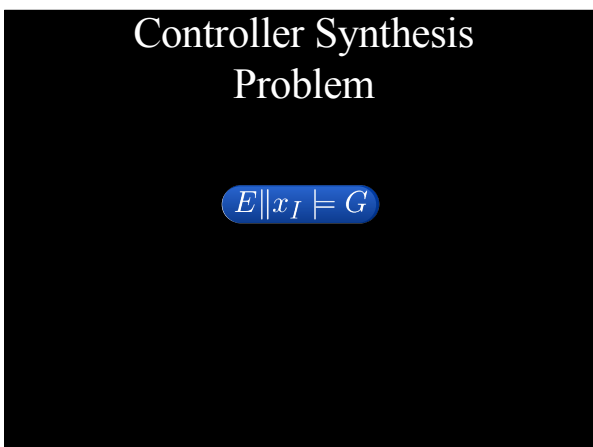
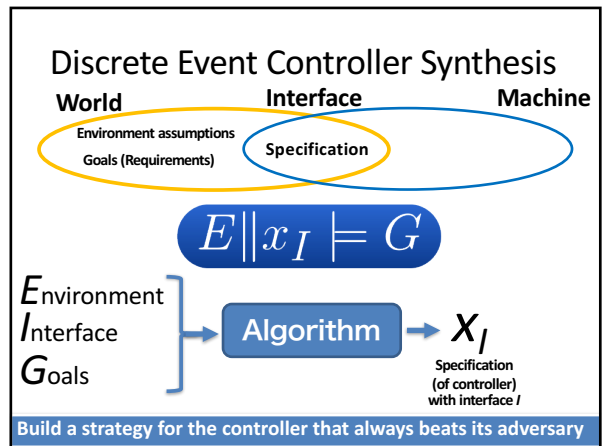
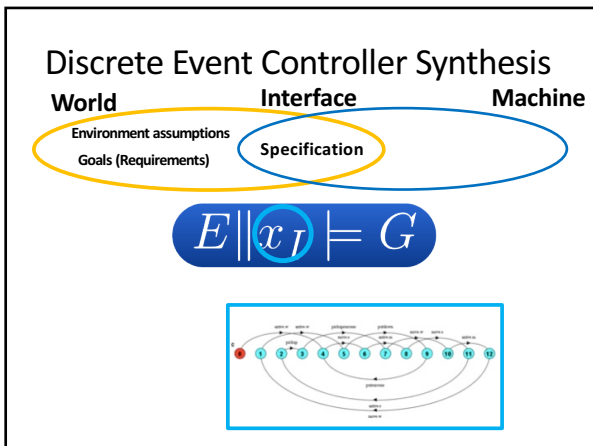
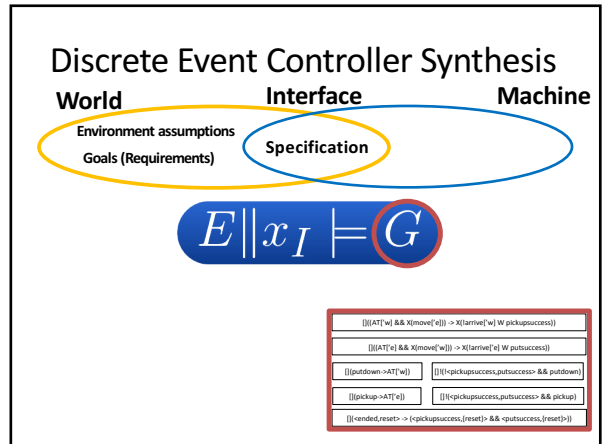
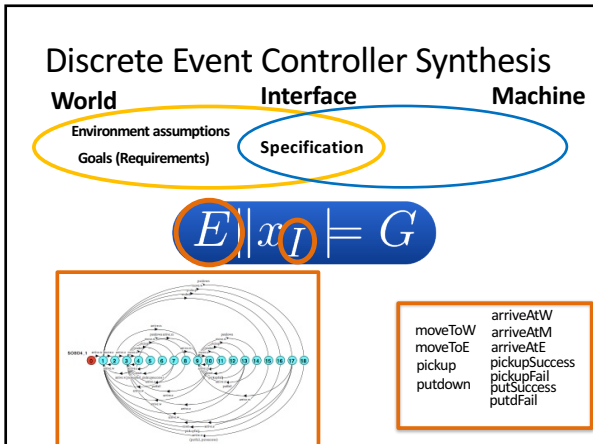
tei@nii.ac.jp

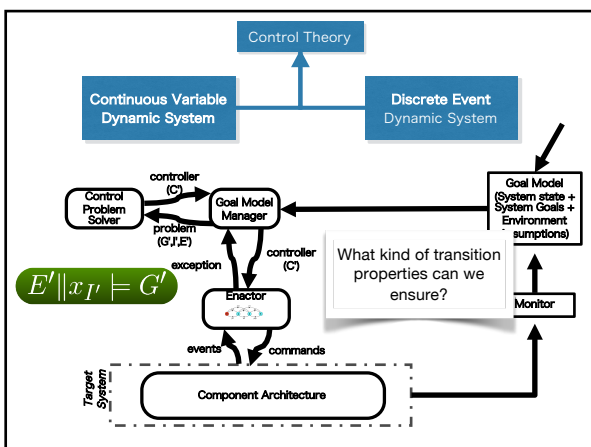
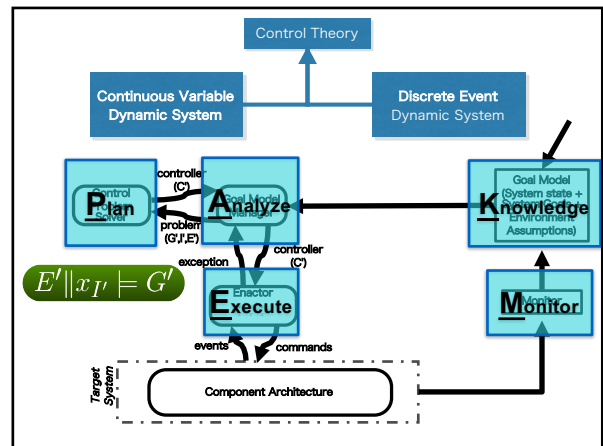
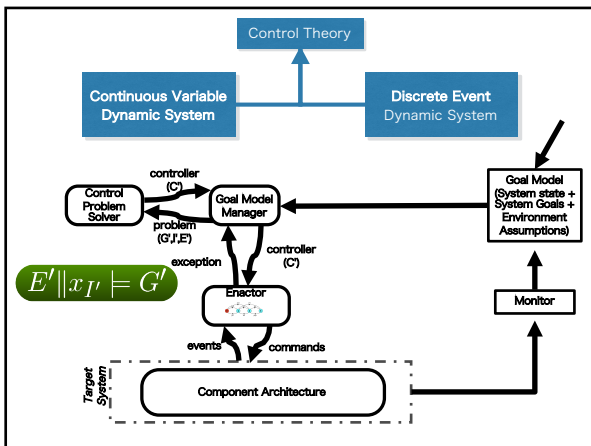
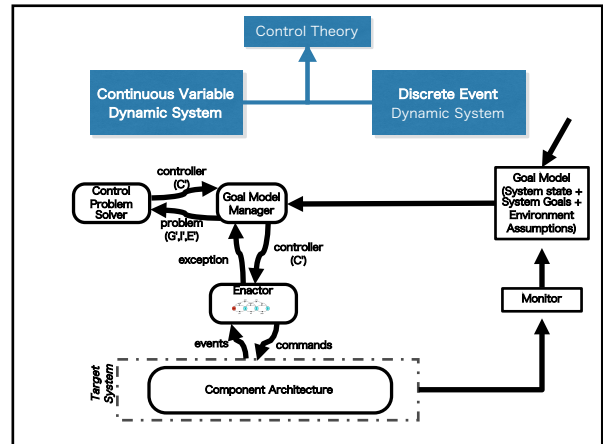
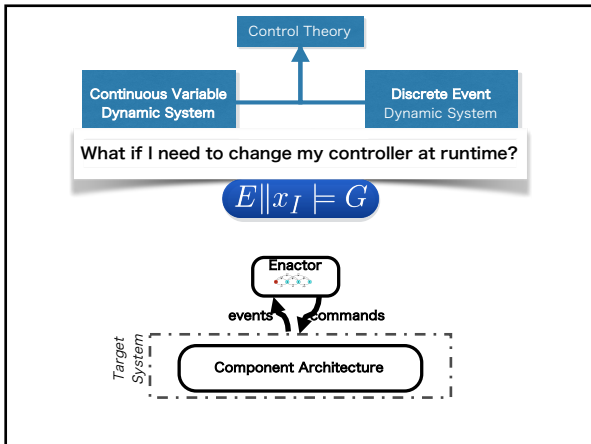
<http://researchmap.jp/teikenji/?lang=english>

In collaboration with Leandro Nahabedian, Victor Braberman, Nicolas D'Ippolito, Shinichi Honiden, Jeff Kramer, and Sebastian Uchitel.

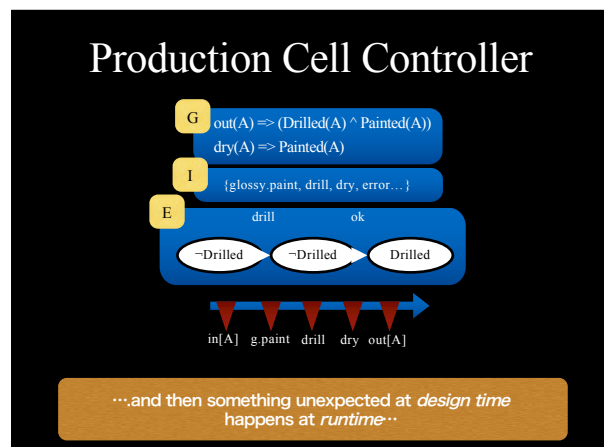
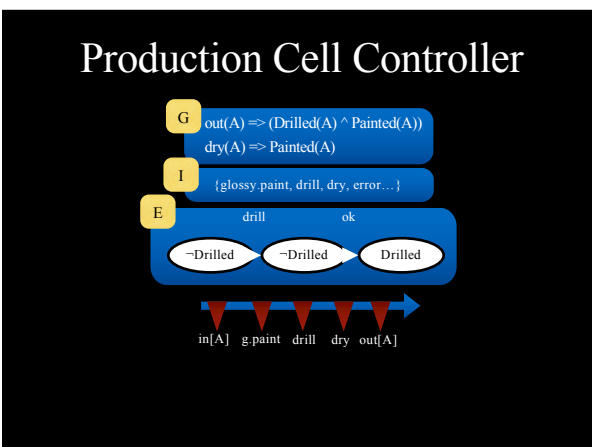
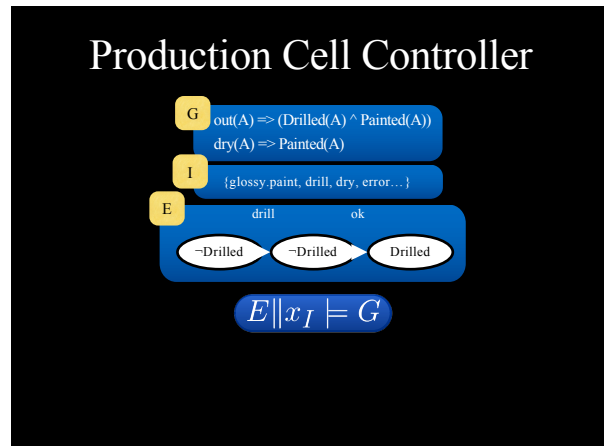
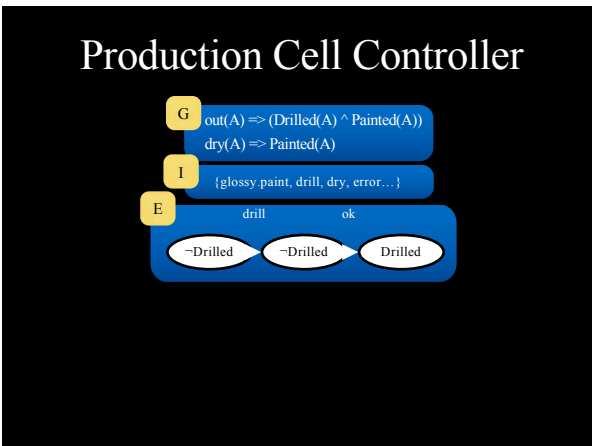
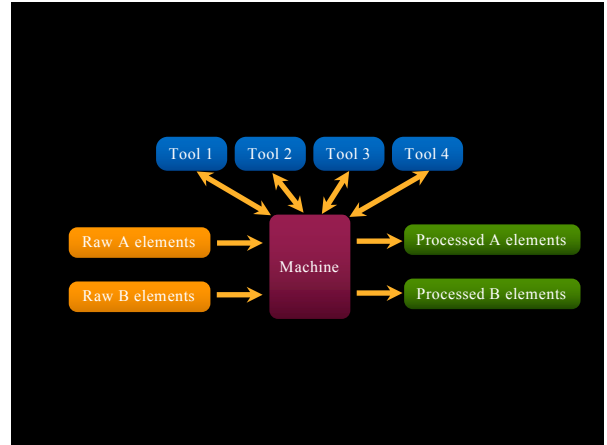
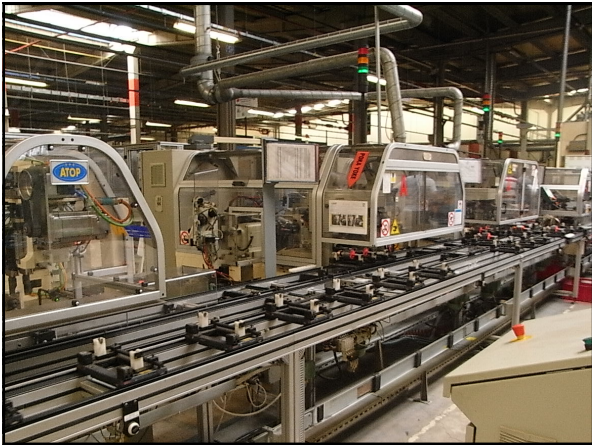
This work was presented at SEAMS 2016 and selected as best paper

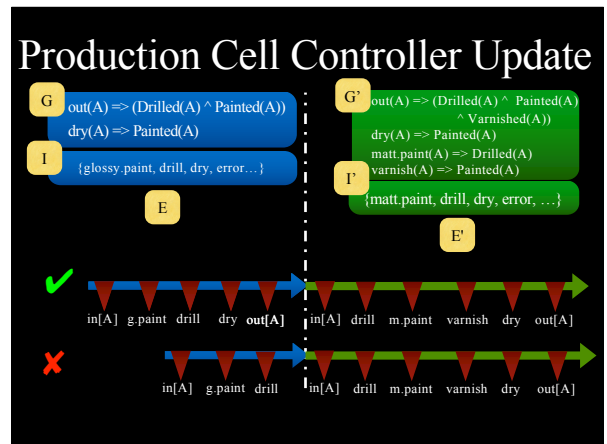
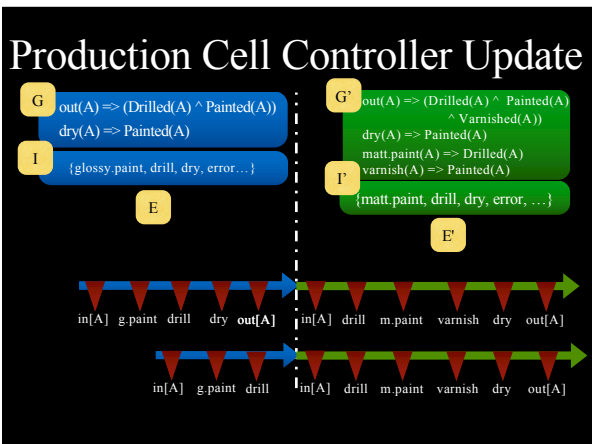
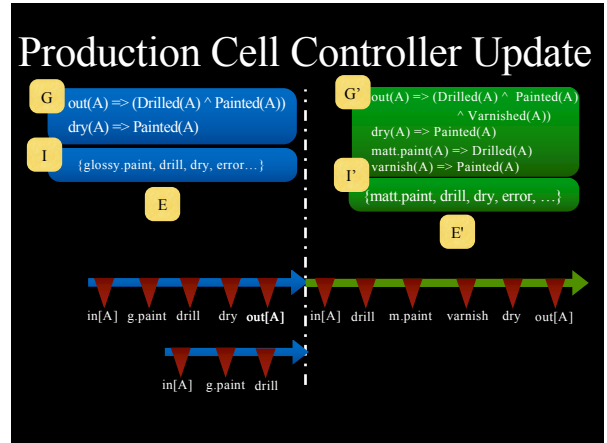
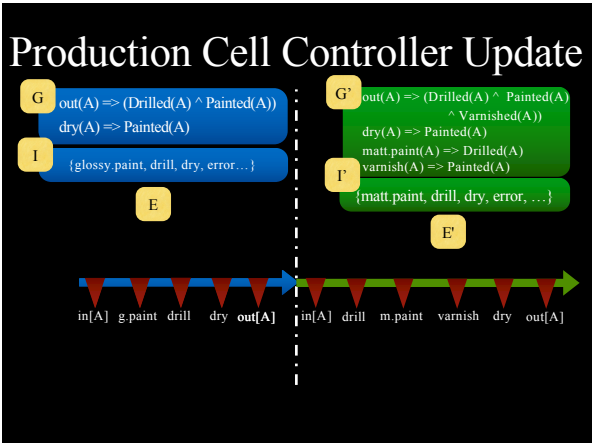
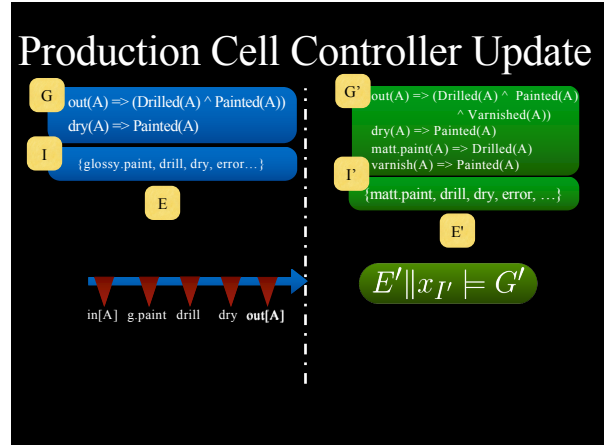
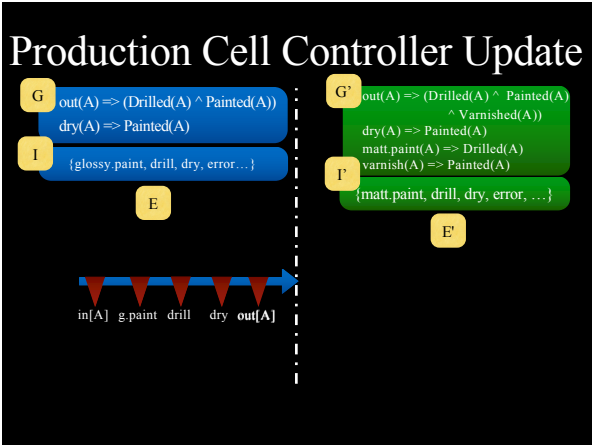




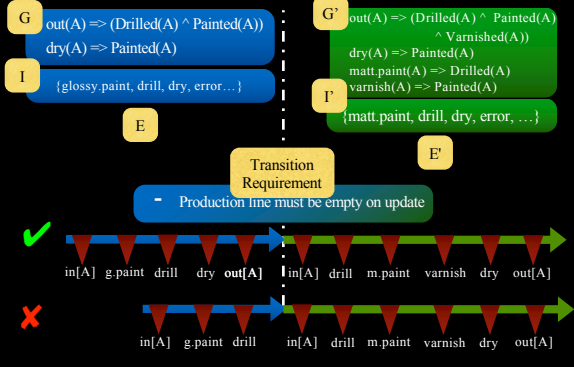


What happens when you change (a discrete event) controller at runtime?



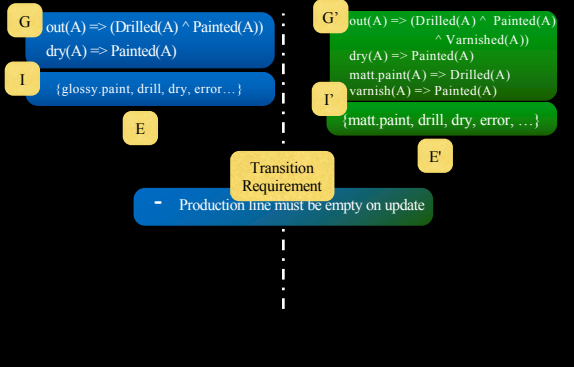


Production Cell Controller Update

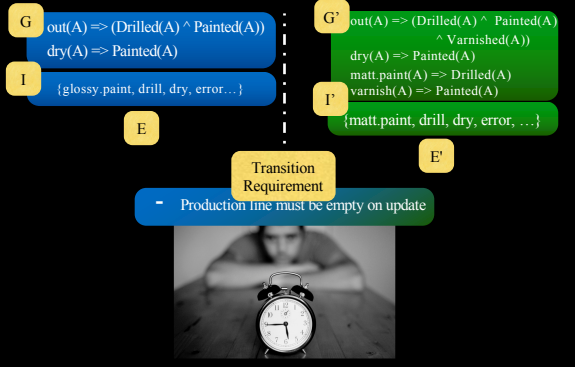


Transition requirements matter!
... and must be explicit.

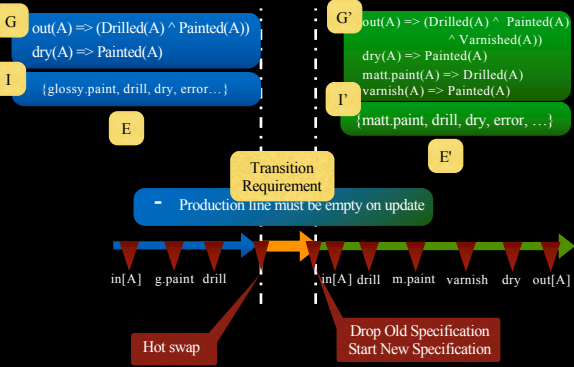
Production Cell Controller Update



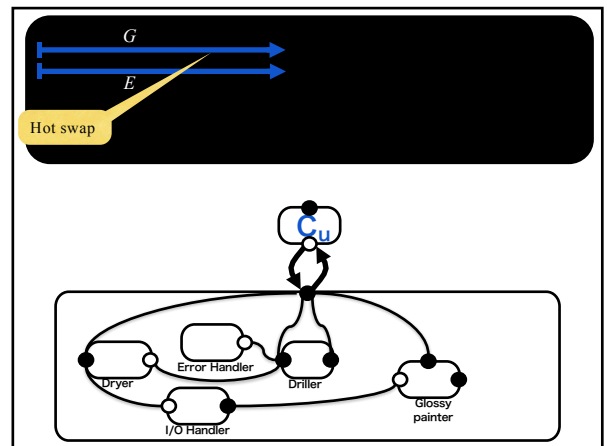
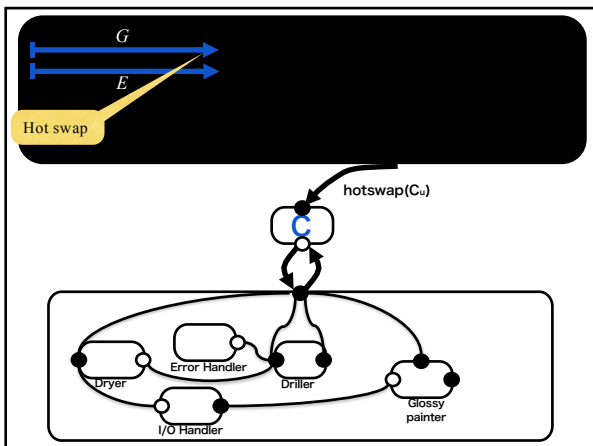
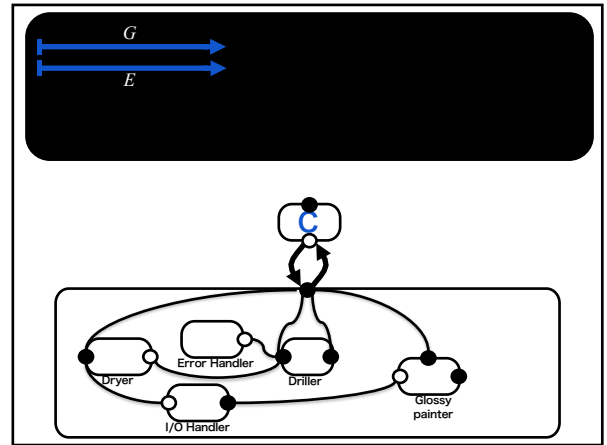
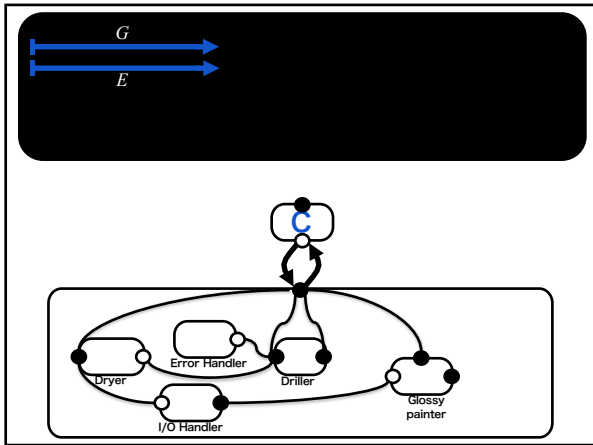
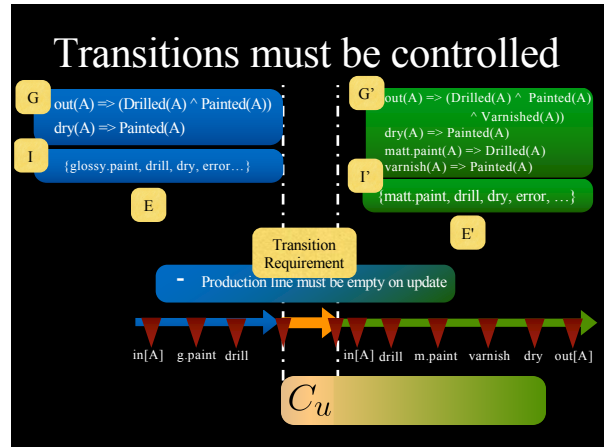
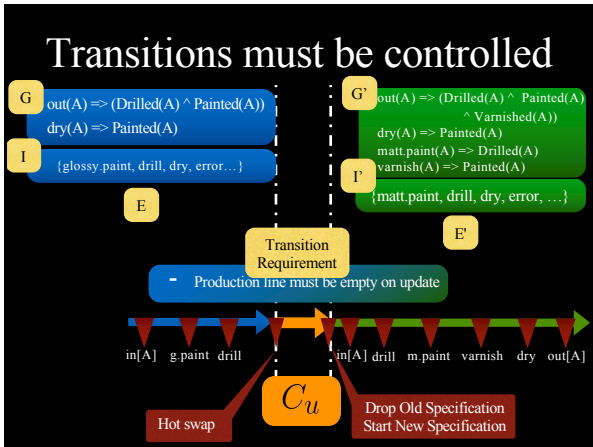
Production Cell Controller Update

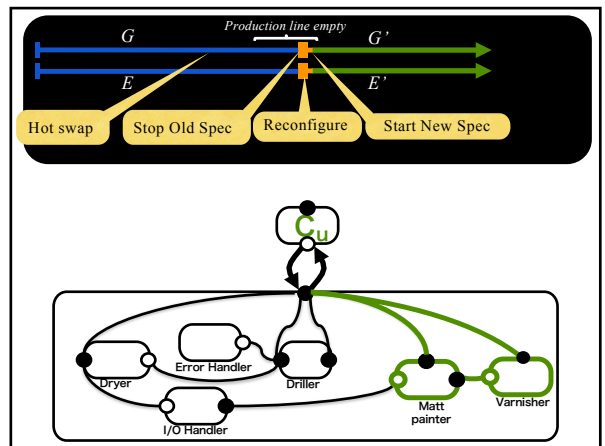
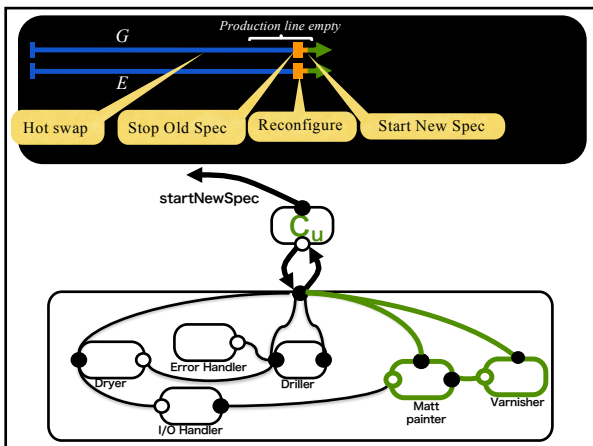
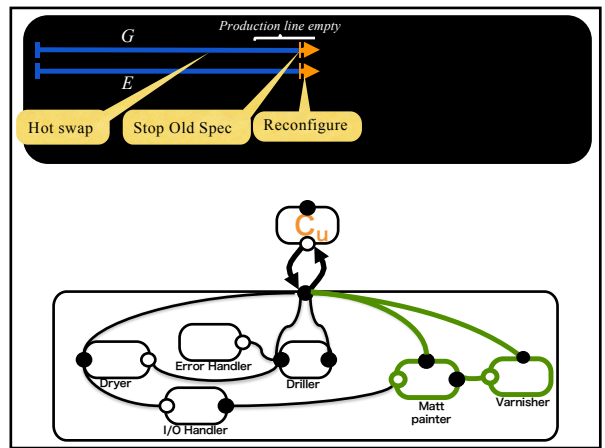
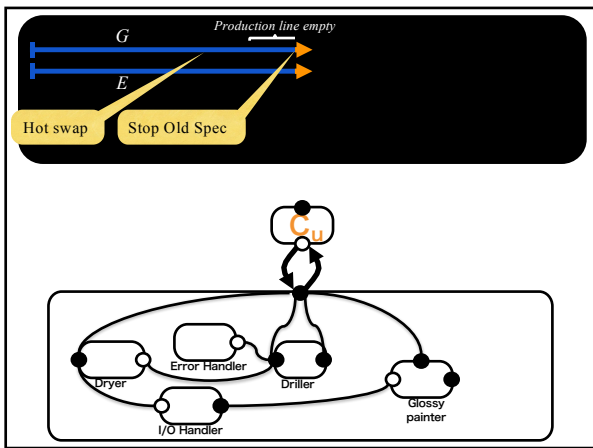
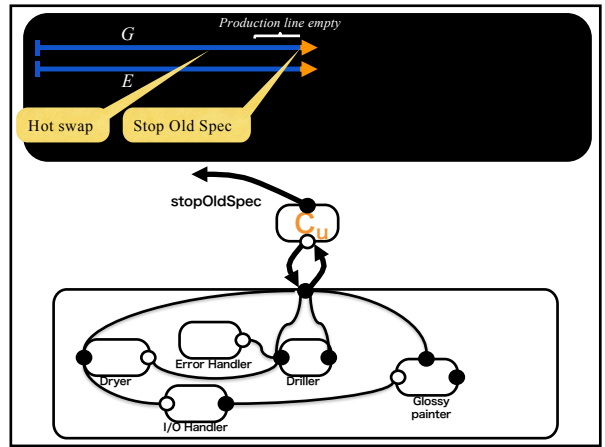
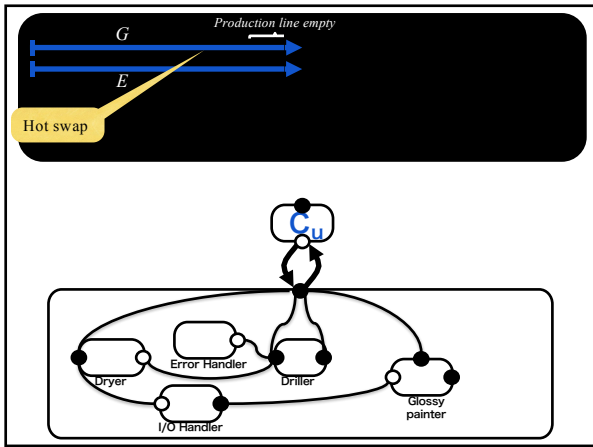


Transitions must be controlled



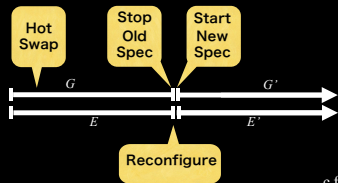
Systems must be guided to safe update states...
...reaching such state should be guaranteed.





Transition Requirements (1/3)

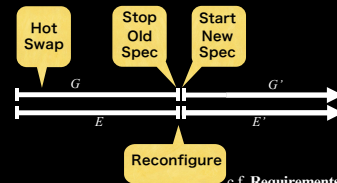
- Production line must be empty on update (StartNewSpec => EmptyProductionLine)



c.f. quiescence

Transition Requirements (2/3)

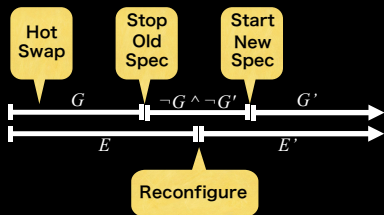
- Production line must not have A elements on update (StoppedOldSpec && !StartedNewSpec => NoAElements)



c.f. Requirements aware quiescence

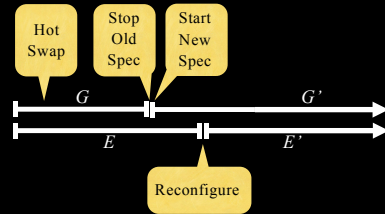
Transition Requirements (3/3)

- Partially processed A elements must be discarded. (InTransition && PartiallyProcessed => Discard before StartNewSpec)



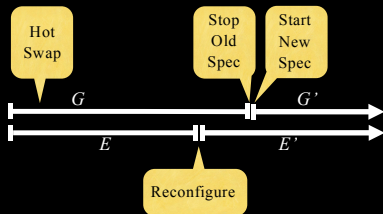
Dynamic Controller Update

- The Full (Abstract) Picture -



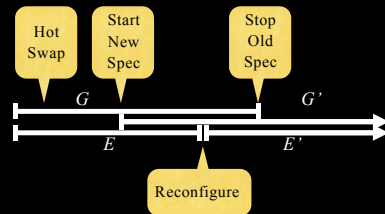
Dynamic Controller Update

- The Full (Abstract) Picture -

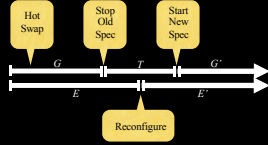


Dynamic Controller Update

- The Full (Abstract) Picture -

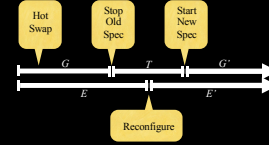


Dynamic Controller Update - The Full (Abstract) Picture -



- T holds
- G holds until StopOldSpec
- G' holds after StartNewSpec
- If HotSwap then StartOldSpec, StartNewSpec and Reconfigure will occur

Dynamic Controller Update - The Full (Abstract) Picture -



- T holds
- G holds until StopOldSpec
- G' holds after StartNewSpec
- If HotSwap then StartOldSpec, StartNewSpec and Reconfigure will occur

How do we build a controller that can do this?

Controller Update Synthesis

Find C_u with interface I_u such that:

$$E_u || C_u \models G_u$$

Goal for Controller Update Synthesis

Find C_u with interface I_u such that:

$$E_u || C_u \models G_u$$

- $$G_u = \left\{ \begin{array}{l} 1. G \mathbf{W} \text{ stopOldSpec} \\ 2. T \\ 3. \Box(\text{startNewSpec} \Rightarrow \Box G') \\ 4. \Box(\text{hotSwap} \Rightarrow (\Diamond \text{stopOldSpec} \wedge \Diamond \text{reconfigure} \wedge \Diamond \text{startNewSpec})) \end{array} \right.$$

Goal for Controller Update Synthesis

Find C_u with interface I_u such that:

$$E_u || C_u \models G_u$$

- $$G_u = \left\{ \begin{array}{l} 1. G \mathbf{W} \text{ stopOldSpec} \\ 2. T \\ 3. \Box(\text{startNewSpec} \Rightarrow \Box G') \\ 4. \Box(\text{hotSwap} \Rightarrow (\Diamond \text{stopOldSpec} \wedge \Diamond \text{reconfigure} \wedge \Diamond \text{startNewSpec})) \end{array} \right.$$
- $I_u = I \cup I' \cup \{\text{hotSwap}, \text{startNewSpec}, \text{stopOldSpec}, \text{reconfig}\}$

Goal for Controller Update Synthesis

Find C_u with interface I_u such that:

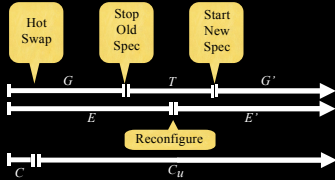
$$E_u || C_u \models G_u$$

- $$G_u = \left\{ \begin{array}{l} 1. G \mathbf{W} \text{ stopOldSpec} \\ 2. T \\ 3. \Box(\text{startNewSpec} \Rightarrow \Box G') \\ 4. \Box(\text{hotSwap} \Rightarrow (\Diamond \text{stopOldSpec} \wedge \Diamond \text{reconfigure} \wedge \Diamond \text{startNewSpec})) \end{array} \right.$$
- $I_u = I \cup I' \cup \{\text{hotSwap}, \text{startNewSpec}, \text{stopOldSpec}, \text{reconfig}\}$

$$E_u = ?$$

Environment for Controller Update Synthesis

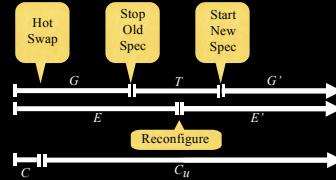
$$E_u || C_u \models G_u$$



$$E_u = ?$$

Environment for Controller Update Synthesis

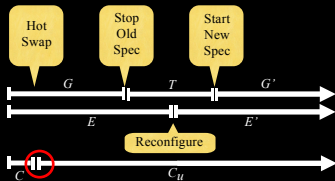
$$E_u || C_u \models G_u$$



$$E_u = E \blacktriangleright reconfig. \blacktriangleright E'$$

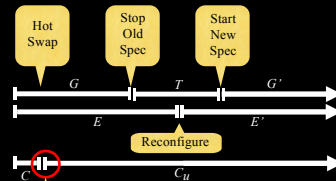
Environment for Controller Update Synthesis

$$E_u || C_u \models G_u$$



Environment for Controller Update Synthesis

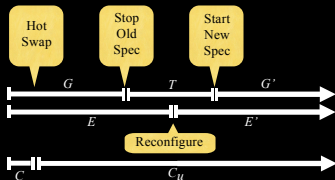
$$E_u || C_u \models G_u$$



Structurally equivalent

Environment for Controller Update Synthesis

$$E_u || C_u \models G_u$$



$$E_u = (E||C) \blacktriangleright hotswap \blacktriangleright E \blacktriangleright reconfig. \blacktriangleright E'$$

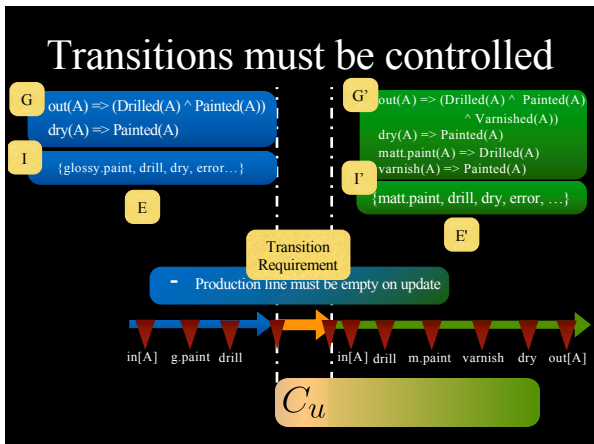
Goal for Controller Update Synthesis

$$E_u || C_u \models G_u$$

$$G_u = \left\{ \begin{array}{l} 1. G \text{ W } stopOldSpec \\ 2. T \\ 3. \square(startNewSpec \implies \square G') \\ 4. \square(hotSwap \implies (\diamond stopOldSpec \wedge \diamond reconfigure \wedge \diamond startNewSpec)) \end{array} \right.$$

$$I_u = I \cup I' \cup \{hotSwap, startNewSpec, stopOldSpec, reconfig.\}$$

$$E_u = (E||C) \blacktriangleright hotswap \blacktriangleright E \blacktriangleright reconfig. \blacktriangleright E'$$



Dynamic Controller Update

- *Model-based development of dynamically adaptive software. Zhang and Cheng. ICSE'16*
- *Specifying adaptation semantics. Zhang and Cheng WADS'05*

Dynamic Controller Update

- *Model-based development of dynamically adaptive software. Zhang and Cheng. ICSE'16*
- *Specifying adaptation semantics. Zhang and Cheng WADS'05*
- *Formalizing Correctness Criteria of Dynamic Updates Derived from Specification Changes. Panzica La Manna, Greenyer, Ghezzi, Brenner. SEAMS'13*
- *Synthesizing Dynamically Updating Controllers from Changes in Scenario-Based Specifications. Ghezzi, Greenyer, Panzica La Manna. SEAMS'12*

Dynamic Controller Update

- **General:** Supports explicit transition requirements and reconfiguration
- **Assured:** System is guaranteed to reach an updatable state
- **Correct:** Transition requirements and new specification are guaranteed by construction
- **Fully automated:** We use controller synthesis



Summary

- Context
 - Environment will change at runtime
 - How do we ensure correctness of software?
=> *Models@run.time approach enables decision making when more information is available*
- Tech. Topic: Update of controller
 - Software (controller) will be changed at runtime in response to changes in the environment
 - How do we ensure that update of controller is correct?
=> *Synthesize updating controller!*

Thank you