Dynamic Evolution by Constraint Orchestration
position paper

Luuk Groenewegen
Leiden University

Formal Models and Notation for BPEx, SOC and CSCW
Copenhagen, October 20, 2007

joint work with Erik de Vink
Outline

- Problem situation
- Solution for Foreseen Coordination
- Solution for Unforeseen Coordination
- Outlook
Problem Situation
composite structure diagram

in usual form

as collaboration diagram
composite structure diagram

in combined form
**dynamic consistency in stable situation**

three types of dynamic consistency problems
dynamic consistency and migration

more types of dynamic consistency problems
**dynamic consistency and migration**

During Original Situation

During Migration

four more dynamic consistency problem types
dynamic consistency and migration

seven dynamic consistency problem types
Solution for Foreseen Coordination
collaboration

critical section management
architecture
and collaboration

three workers and one scheduler
local processes

worker / scheduler STDs
both of type A dynamics
constraints and constraint composition

protocol role of worker: type B dynamics
subprocesses checked, enforcing traps
protocol as constraint orchestration

traps checked, enforcing subprocesses synchronized roles: type D dynamics
consistency rules

Scheduler : \( \text{Check}_i \xrightarrow{\text{allow}} \text{Asg}_i \) * \\
\( \text{Worker}_i(\text{CSM}) : \text{OutBlock} \xrightarrow{\text{started}} \text{InCS} \)

Scheduler : \( \text{Asg}_i \xrightarrow{\text{disallow}} \text{Check}_{i+1} \) * \\
\( \text{Worker}_i(\text{CSM}) : \text{InCS} \xrightarrow{\text{done}} \text{OutCS} \), \\
\( \text{Worker}_{i+1}(\text{CSM}) : \text{OutCS} \xrightarrow{\text{triv}} \text{OutBlock} \)

Scheduler : \( \text{Check}_i \xrightarrow{\text{continue}} \text{Check}_{i+1} \) * \\
\( \text{Worker}_i(\text{CSM}) : \text{OutBlock} \xrightarrow{\text{notYet}} \text{OutCS} \), \\
\( \text{Worker}_{i+1}(\text{CSM}) : \text{OutCS} \xrightarrow{\text{triv}} \text{OutBlock} \)
Solution for Unforeseen Coordination
McPal, stand-by

during stable collaboration, when no migration occurs
McPal’s evolutionary constraints and protocol role partly unknown

alternating between foreseen and unforeseen phase
collaboration for evolution

McPal as manager of all components
consistency rules for McPal

\[\text{McPal: Observing} \xrightarrow{\text{wantChange}} \text{JITting}\]

\[\text{McPal: JITting} \xrightarrow{\text{knowChange}} \text{NewRuleSet} \ast\]

\[\text{McPal[} \text{Crs} : = \text{Crs} + \text{Crs}_{\text{migr}} + \text{Crs}_{\text{toBe}} \text{]}\]
consistency rules for McPal

\[ \text{McPal}: \text{Observing} \xrightarrow{\text{wantChange}} \text{JITting} \]

\[ \text{McPal}: \text{JITting} \xrightarrow{\text{knowChange}} \text{NewRuleSet} * \]
\[ \text{McPal}[\text{Crs} : = \text{Crs} + \text{Crs}_{migr} + \text{Crs}_{toBe}] \]

\[ \text{McPal}: \text{NewRuleSet} \xrightarrow{\text{giveOut}} \text{StartMigr} * \]
\[ \text{McPal}(\text{Evol}): \text{StatPhase} \xrightarrow{\text{ready}} \text{MigrPhase} \]

\[ \text{McPal}: \text{Content} \xrightarrow{\text{phaseOut}} \text{Observing} * \]
\[ \text{McPal}(\text{Evol}): \text{MigrPhase} \xrightarrow{\text{migrDone}} \text{StatPhase}, \]
\[ \text{McPal}[\text{Crs} : = \text{Crs}_{toBe}] \]
migration coordination as constraint orchestration

Luuk Groenewegen & Erik de Vink

Dynamic Evolution by Constraint Orchestration
Outlook
concluding remarks

- Paradigm addresses coordination: T1 – T3
- McPal addresses migration: T4 – T7
- business and software dynamics treated alike
concluding remarks

- process vs service oriented: collaborations forged
- migration without quiescence: on-the-fly
- migration is consistency-conserving: repeatable
- emerging collaboration dynamics: deployable
- future work
  aspect weaving, co-evolution, ArchiMate-UML,
  reorganization patterns, tool support