Collective Adaptive Process-Aware Information Systems: Challenges, Scenarios, Techniques

Manfred Reichert
Agenda

Adaptive Collective Systems

Backgrounds

Adaptive Collective Object Lifecycles

Adaptive Cross-Organizational Processes
Adaptive Collective Systems

• “Heterogeneous collections of autonomous task-oriented systems that cooperate on common goals forming a collective system…”

• Purpose (cf. FoCAS):
  “The socio-technical fabric of our society more and more depends on systems that are constructed as a collective of heterogeneous components and that are tightly entangled with humans and social structures. Their components increasingly need to be able to evolve, collaborate and function as a part of an artificial society.”
Adaptive Collective Systems

Relevant issues …

• **Nature**  
  *A strong orientation to working the way that natural systems work.*

• **Automated or Facilitated?**  
  *Some of the research oriented toward facilitating humans in an organization, and some is toward replacing humans with automated, yet flexible, systems.*

• **Non-Uniform**
  - *There may be many diverse components interacting in complex ways, i.e., it is not assumed that there is a single uniform process system*
  - *Diversity is the important ingredient for stability in the face of unexpected changes.*

Source: K. Swenson, 2014
Agenda

- Adaptive Collective Systems
- Backgrounds
- Adaptive Collective Object Lifecycles
- Adaptive Cross-Organizational Processes
Process-Aware Information Systems (PAIS)

Process Schema S

- Patient Admission
- Anamnesis & Clinical Examination
- X-ray
- MRT
- Sonography
- Non Operative Therapy
- Initial Treatment & Operation Planning
- Operative Treatment
- Discharge & Documentation

Activity States:
- ▲ Activated
- ✔ Completed
- ✗ Skipped

Execution Trace:
σ₁ = < „Patient Admission“, „Anamnesis & Clinical Examination“, „X-ray“>

σ₂ = < „Patient Admission“, „Anamnesis & Clinical Examination“, „Non Operative Therapy“>
Process-Aware Information Systems (PAIS)

**Process-aware Information System (PAIS)**

- Late Modeling
- Web Client API
- Admin. API
- Message Queuing
- Process Execution Engine
- Model Event
- Process Composition
- Process Execution
- Local Control
- Service Development
- User Interface

**Process Composer**
- Create Process Schema
- Modify Process Schema
- Check Process Schema

**Process Repository**
- Application Components
- Process Schemas

**Process Engineer**
Adaptive PAIS: Instance Changes

ADEPT

Individually adaptable Process Instances
Adaptive PAIS: Instance Changes

**ADEPT**

Individually adaptable Process Instances

Achievements:
- Formal and expressive process meta model
- Formal Criteria for Change Correctness
- Efficient, build-in consistency checks ("no bad surprise")
- Support of a high number of change patterns
- API for accomplishing ad-hoc changes
Adaptive PAIS: Schema Evolution

ADEPT Process Management System

- Std Client API
- Web Client API
- Modeling API
- Dynamic Change API
- Admin. API
- Role Management
- Authorization
- Time Management
- Message Queueing
- Recovery
- Audit Trail
- ...
Adaptive PAIS: Extended Process Lifecycle

1. Create Process Schema
2. Create Instances
3. Process Execution
4. Process Monitoring
5. Evolve Process Schema
6. Process participant
7. Change Propagation

Exception: Delete (I₁, E)
Transferring ADEPT to Industrial Practice
Agenda

Adaptive Collective Systems

Backgrounds

Adaptive Collective Object Lifecycles

Adaptive Cross-Organizational Processes
Adaptive Collective Object Lifecycles
Adaptive Collective Object Lifecycles
Adaptive Collective Object Lifecycles: Drivers
The Corepro Framework: Overview

Modellebene

Datenmodell

Data Model

Life Cycle Coordination Model

Object Life Cycles / Life Cycle Coordination Model

Instanzebene

Datenstruktur

Data Structure

Data-driven Process Structure
The Corepro Framework: Overview

**Modellebene**

- Gesamtsystem
  - hatSys
    - System
      - hatKomp
        - nutztKomp
          - Komponente

**Instanzebene**

- Gesamtsystem
  - BR212, Rel. 02/08
    - hatSys
      - System
        - Telematik High V2.2
          - hatKomp
            - nutztKomp
              - Komponente
                - Head-Up Unit V3.14
              - nutztKomp
                - Komponente
                  - TV Tuner V1.83

- Datenstruktur

- Datengetriebene Prozessstruktur

Object Life Cycles / Life Cycle Coordination Model

Data-driven Process Structure
The Corepro Framework: Overview

Data-driven Process Structure
The Corepro Framework: Overview
• Significant reduction of modeling efforts
• Formal semantics enables correct executability
The Corepro Framework: Data-Driven Adaptation

Change Operation (Data Structure)
1) removeRelation(Telematik High V2.2, TV Tuner V1.83, nutztKomp);
2) removeObject(TV Tuner V1.83);

Change Operation (Process Structure)
1) removeExtTrans(Telematik High V2.2, Muster Aufgebaut, Installieren, TV Tuner V1.83 . E);
2) removeOLC(Tuner V1.83);
The Corepro Framework: Exception Handling

Forward Recovery

1. Step 1
2. Step 2
3. Step 3

The Corepro Framework: Exception Handling
The Corepro Framework: Proof-of-Concept

Image of a graphical user interface showing process structures and states such as 'Not Activated', 'Activated', 'Done', 'Skipped', 'Inkonsistenz erkannt', 'Waiting', 'Processing', 'Fired', 'Disabled', 'Zustandsmarkierungen', and 'Transitionsmarkierungen'.
The Corepro Framework: Case Study

Instance Level: Data Safety - Sub-systems System Level: Process Structure

**System**

- **Sub-system A**
  - 4-5: Specification of technical safety concept
  - 4-6: System design
  - Part 5: PD HW
  - Part 6: PD SW
  - 4-7.4.1: HW-/SW integration and tests

- **Sub-system B**
  - 4-5: Specification of technical safety concept
  - 4-6: System design
  - Part 5: PD HW
  - Part 6: PD SW

- **Sub-system B1**
  - 4-5: Specification of technical safety concept
  - 4-6: System design
  - Part 5: PD HW
  - Part 6: PD SW
  - 4-7.4.1: HW-/SW integration and tests

- **Sub-system B2**
  - 4-5: Specification of technical safety concept
  - 4-6: System design
  - Part 5: PD HW
  - Part 6: PD SW
  - 4-7.4.1: HW-/SW integration and tests

- **System**
  - 4-4: Initiation of product development at system level
  - 4-5: Specification of technical safety concept
  - 4-6: System design

**PD** = Product Development, **HW** = Hardware, **SW** = Software

ISO 26262:
Road Vehicles, Functional Safety
Agenda

Adaptive Collective Systems

Backgrounds

Adaptive Collective Object Lifecycles

Adaptive Cross-Organizational Processes
Adaptive Cross-Organizational Processes

Cross-Organizational Processes

Business Process Compliance

Adaptation & Evolution

Process-aware Information System

Running Instances / Process Engine

Process Models

Process Logs
Modeling Cross-Organizational Processes: Interaction Modeling
Cross-Organizational Processes
Cross-Organizational Processes
Cross-Organizational Processes: Interconnection Modeling
Cross-Organizational Processes: Interaction Modeling
Interaction Modeling:
Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DAChor

\[
A ::= BA \quad \text{(basic activities)} \\
| A; A \quad \text{(sequential composition)} \\
| A \sqcap A \quad \text{(choice)} \\
| A \parallel A \quad \text{(parallel composition)}
\]

\[
BA ::= \text{skip} \quad \text{(no action)} \\
| a_i \quad \text{(activity in role } R_i) \\
| c[i,j] \quad \text{(communication)}
\]

**Basic:**
- \([\text{skip}] \equiv \emptyset\)
- \([a] \equiv \{a\}\)
- \([c[i,j]] \equiv \{c[i,j]\}\)

**Sequential:**
- \([A_1; A_2] \equiv [A_1] \cap [A_2]\)

**Choice:**
- \([A_1 \sqcap A_2] \equiv [A_1] \cup [A_2]\)

**Parallel:**
- \([A_1 \parallel A_2] \equiv \text{interlv}([A_1], [A_2])\)

Qiu Zongyan, Zhao Xiangpeng, Cai Chao, and Yang Hongli:
Towards the Theoretical Foundation of Choreography, WWW’07, 2007.
Interaction Modeling:
Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACHor


Interaction Modeling:
Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACChor

Gero Decker, Mathias Weske:
Interaction Modeling:
Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DAChor

Johannes Maria Zaha, Alistair Barros, Marlon Dumas, Arthur ter Hofstede:
Interaction Modeling: Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let's Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACChor

**Interaction Modeling:**

**Languages, Notations, Standards**

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACChor

W3C: Web services choreography description language 1.0, 2005.
Interaction Modeling: Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACHor


What about Data?
Interaction Modeling:
Languages, Notations, Standards

- Chor
- Conversation protocols
- Interaction Petri-Nets
- Let’s Dance
- iBPMN
- WS-CDL
- BPMN 2.0 Choreography
- DACChor
Ensuring Compliance of Cross-Organizational Processes


Cross-Organizational Process Compliance

Business Process Compliance

Cross-Organizational Processes

Change / Flexibility

C³Pro

Interaction Modeling

Analysis

Execution

Change
Scenario: Electronic Change Request

**interaction model**
- Change team
  - Inform about ECR
    - Engineers
  - Request comments
    - Engineers
  - Analyst
    - Change details
    - Change team, engineers
  - Decision team
    - Approval
    - Change team, engineers

**local views** – e.g. local view of engineering team
- Change team
  - Inform about ECR
    - Engineers
  - Request comments
    - Engineers
  - Analyst
    - Change details
    - Change team, engineers
  - Decision team
    - Approval
    - Change team, engineers

**private process models**
- Analyst
  - Request change details
  - Study change request
  - Determine involved components
  - Detail change
  - Change details

- Change team
  - Specify ECR
  - Request comments
    - Comments on ECR
  - Review & include comments
  - Request change details
  - Change details
  - Prepare final ECR
  - Request approval
  - Approval
  - Document ECR

- Analyst
  - Inform about ECR
  - Suspend affected development tasks
  - Request comments
  - Analyze ECR
  - Determine side effects
  - Write comments
  - Comments on ECR
  - Change details
  - Approval
  - Implement change & continue development tasks
Relevant Layers

interaction model

local views

public process models

private process models

Public

Private
The Process of Interaction Modeling

1. Specify Interaction Model
2. Compute Local Views
3. Specify Public Process Models
4. Specify Private Process Models
5. Check Realizability
6. Revise Interaction Model

Steps:
- Inform about ECR
- Request comments
- Write comments
- Analyze ECR
- Determine side effects
- Implement change

Decision Points:
- Realizable?
- Change team
- Change details
- Approval
Relevant Layers and Compliance Rules

Interaction Model

Public
- change team
- inform about ECR
- engineers

Private
- change team
- request comments
- engineers

Local Views
- change team
- inform about ECR
- engineers

Public Process Models
- inform about ECR
- request comments

Private Process Models
- inform about ECR
- suspend affected development tasks
- request comments
- analyse ECR
- determine side effects
- write comments
- change details
- approval
- implement change & continue develop. tasks

Affected development tasks must be suspended before implementing changes.
The engineers must determine side effects before the decision team decides on the change.
Relevant Layers and Compliance Rules

Side effects must be determined before sending comments on the ECR to the change team.
The Process of Interaction Modeling

1. Select/specify compliance rules
2. Specify public process models & assertions
3. Specify private process models
4. Ensure local & asserted compliance
5. Specify interaction model
6. Compute local views
7. Check realizability
8. Revise public process models & assertions
9. Revise interaction model
Comments on the ECR must not be requested before change details are sent to the engineers.
The Process of Interaction Modeling

1. Select/specify compliance rules
2. Specify public process models & assertions
3. Check global compliance
4. Compliant?
5. Specify private process models
6. Specify interaction model
7. Compute local views
8. Check realizability
9. Realizable?
10. Revise public process models & assertions
11. Ensure local & asserted compliance
12. Release
13. Check realizability & compliability
14. Revise interaction model
Relevant Layers and Compliance Rules

Public

- interaction model
  - change team
    - inform about ECR
    - engineers
  - change team
    - request comments
    - engineers
  - engineers
    - comments on ECR
    - change team
  - change team
    - request change details
    - analyst
  - analyst
    - change details
    - change team, engineers
  - change team
    - request approval
    - decision team
  - decision team
    - approval
    - change team, engineers

local views

- change team
  - inform about ECR
  - engineers
- change team
  - request comments
  - engineers
- engineers
  - comments on ECR
  - change team
- analyst
  - change details
  - change team, engineers
- decision team
  - approval
  - change team, engineers

Public process models

- inform about ECR
- request comments
- write comments
- comments on ECR
- change details
- approval
- implement change

Private process models

- inform about ECR
- Suspend affected development tasks
- request comments
- analyse ECR
- determine side effects
- write comments
- comments on ECR
- change details
- approval
- implement change & continue developing tasks

Private

- local compliance rules incl. assertions
- compliance rules
- global compliance rules
- compliability?
Enabling Adaptations and Changes of Cross-Organizational Processes

Global Choreography Model (Interaction Model)

Model Abstraction

Public View of Partner B
Basics
Public View of Partner B

Private View of Partner B

Public View

Conformant with?

Private View

Private activities, e.g., data transformation

Basics
Global Schema

Basics

Global Choreography Model

Compatibility

Consistency
- Ok! and what is the problem then?

- Conformance (Consistency)
- Behavioral Compatibility
  - Waiting for a message which will never arrive
  - Sending message which will not be consumed
- Structural compatibility
- Transitivity effects
- Negotiation
Change Propagation

Choreography Model
  + Public views
  + Change Specification

Partners affected by the change
  + Changes to be propagated

- Preserve Conformance
- Preserve Compatibility
- Transitive effects
- Negotiation 🙅‍♂️ 😞
Change Propagation: Negotiation

Start
  Specify change
  Infer interaction changes
  Variant?
    Yes
    Compute affected partners
  No
    Update local change
  Find another alternative
    Succeed
    Update global choreography model
    Update local choreography models
    Succeed
    Compute public2private changes
    Yes
    Negotiate changes
    last partner?
    Yes
    Negotiate changes
    Compute changes to propagate to this partner
    No
    Compute public2private changes
    No
    negociations succeed?
    Yes
    Negotiate changes
    No
    abandon?
    Succeed
    End
    Fail
    ok?
    Yes
    Check compatibility and consistency
    No
    Propagate changes public2private
    Succeed
    Update global choreography model
    Fail
    Update local choreography models
Change Propagation: Change Patterns

- **INSERT(fragment, how, in, out)**
  - inserts a new fragment in a process model.

- **DELETE(fragment)**
  - Deletes an existing fragment from a process model.

- **REPLACE(oldFragment, newFragment)**
  - Replaces an existing fragment by a new one in the process model.

- **UPDATE(activity, attribute, newValue)**
  - Updates the attributes of a single activity of a process model.
  - *Attribute could be: partner, role, input, output, etc.*
**Change Propagation: Replace Pattern**

**Public View of Partner B**

- A --> B
- B --> C
- C --> B
- E --> B
- XOR

**Private View of Partner B**

- Receive A
- Send C
- XOR
- Receive C
- a2
- AND
- Receive E
- a3
- AND
- a4
- XOR

Conforming with?
Change Propagation: Replace Pattern
Change Propagation: Replace Pattern
Change Propagation: Replace Pattern

Private View

Public View

Abstraction

Reduction Rules
Change Propagation: Replace Pattern

Private View
- Receive A
  - a1
  - Send C
  - XOR
- Receive C
  - a2
  - XOR
  - a4

Public View
- Send C
- A→B
- B→C
- C→B
- E→B

Abstraction
Reduction Rules
Invariant!
Change Propagation: Replace Pattern
Change Propagation: Replace Pattern

Private View

Public View

Abstraction

Reduction rules
Change Propagation: Replace Pattern

**Private View**
- Receive A
  - a1
  - Send C
- Receive C
  - a2
- XOR
- Receive E
  - a3
- AND
- XOR
  - a4

**Public View**
- Send E
- XOR
- Receive E
- Send C
- XOR
- XOR
- XOR
- XOR
- XOR

**Abstraction**
- **REPLACE_{PB}(newF, oldF)**
- **VARIANT!**
- **--> REPLACE_{LB}(abstr(newF), abstr(oldF))**
Change Propagation: Replace Pattern

\[ \text{\textit{REPLACE}_{LB}(\text{abstr(newF)}, \text{abstr(oldF)})} \]

Direct partners affected by the change

\[ \{E, C\} \]
Change Propagation: Replace Pattern

```
insert \rightarrow \text{INSERT}_{LB}(F1, \#, \#, \#)  
replace \rightarrow \text{REPLACE}_{LC}(F2, \#)
```

Direct partners affected by the change

\{E, C\}

Abstraction + reduction rules
Change Propagation: Replace Pattern

Public View of Partner B

Public View of C

Public View of E

Global Choreography Model
Change Propagation: Replace Pattern

Global Choreography Model

Public View of Partner B

\[ \rightarrow \text{REPLACE}_L \{ F_2, ? \} \]

Public View of C

\[ \rightarrow \text{INSERT}_L \{ F_1, ?, ?, ?, ? \} \]

Public View of E
Change Propagation: Replace Pattern

Public View of Partner B

\[ \rightarrow \text{REPLACE}_{LC}(F_2, ?) \]

Public View of C

\[ F_2 \]

Public View of E

\[ \rightarrow \text{INSERT}_{LE}(F_1, ?, ?, ?, ?, ?) \]

Global Choreography Model
Change Propagation: Replace Pattern

Public View of Partner B

Public View of E

Global Choreography Model
Any Questions

Change and Compliance for Collaborative Processes