Reconfiguration from First Principles

Markus Stumptner
Advanced Computing Research Centre
University of South Australia
mst@cs.unisa.edu.au

Reconfiguration from First Principles

with a fair bit of pragmatism in the mix

Markus Stumptner
Advanced Computing Research Centre
University of South Australia
mst@cs.unisa.edu.au

…work with Wolfgang Mayer, Arndt Muehlenfeld, Franz Maier, Rajesh Thiagarajan, Michael Schrefl

Overview

- Reconfiguration – a problem in search of a solution?
- What's been done
  - The Configuration View
  - The Diagnosis View
- Why our interest: 2 applications
- Other approaches
- Summary

The Manhart view of Reconfiguration

- Technical configuration, not sales configuration
- Not all implications of technical changes recorded
- Scaleability of approaches
- Need to adapt processes
- "Nobody has really looked at the problem"

The Configuration View

- Conceptual Reconfiguratoin Framework [Männistö et al.]
- ConBaCon [John, Geske]
- Heuristic approaches [Schenner et al.]
- Knowledge aging [Kreuz, Roller]
- VT
- Plakon

Conceptual Reconfiguration Model

[Männistö/Soininen/Tiihonen,Sulonen CWS 99]
[Männistö 02]

- Standard configuration definition
- Change requirements
- Explicit reconfiguration operators with value function for individual cases
- A reconfiguration is a sequence of changes that is optimal according to the value function
- Demonstrated on case study
  - Case company had predefined reconfiguration operations

Reconfiguration in ConBaCon

[John, Geske CSW 00]
[John 02]

- Object-oriented constraint model with explicit aggregation (subobject) relationships
- When reconfiguring, the existing configuration is put in place as weak constraint, whole problem solved as a partial CSP
- No dynamicity, i.e., set of replacement objects must be explicitly listed, deletion of objects (and subobjects occurs as necessary)

Knowledge Aging/Heuristic Repair

- Knowledge Aging [Kreuz,Roller CWS 99]
  - Applies decreasing utility to older KB elements
  - Alternates single component removal/configuration step
- Heuristic Repair [Schenner, Falkner CWS 02] [Schenner, Fleischanderl CWS 03]
  - Greedy repair algorithm
  - Selected catalog of heuristic repair steps (e.g., for solving cardinality violations in relations) operating directly on object structure
  - Limitations on object creation in repair actions (e.g., by limitations on object types)

VT

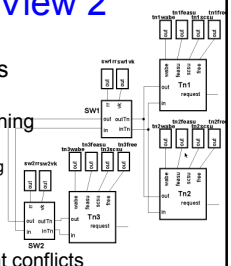
- Elevator design
- "immortalized" in the KE community by the Sisyphus project
  - "Configuration ≅ Propose + revise"
  - "Configuration is a knowledge intensive task"
- Individual "fixes" in case of constraint violations prepared by experts
  - ~40 in the case
  - Potential of "thrashing"
- Not really reconfiguration (since fixing happens within a single search), but very influential in terms of method

The Diagnosis View 1

- Reconfiguration as type of repair method [Crow/Rushby AAAI 91]
  - Triggered in case of explicit misbehavior
  - Specification of fault modes or other behaviour modes
  - No structural modification

The Diagnosis View 2

- Consistency-based diagnosis [Stumptner/Wotawa AiD 98]
  - Example domain: phone switching systems
    - resetting functionality and dealing with configuration errors
  - Standard hitting set algorithm
  - valid reparameterisations
  - structurally minimal component conflicts (removals)
  - no full reconfiguration



A CSP based reconfiguration model

- Based on COCOS OO scheme
- Similar to ConBaCon + dynamics
- Object oriented configuration model
- Costs for constraint changes/additions associated with a solution
  - In terms of assignments/unassignments

Algorithmic considerations

- Three major families
  - CSOP (COP)
  - Explicit reconfiguration planning

Application 1 – Sales Quotation

- Company in automotive sector
- Negotiation process with customers
  - Currently spreadsheet-based
- "standard" configuration aspects
- Reconfiguration in the negotiation process
  - "change of target with time limit
  - Implicit initial assumptions may be lost
- Case-based approach (?)

How does the Sales Quotation project fit the scheme?

- Dependency management
- Keeping track of all the relevant factors
- SW/HW interface??

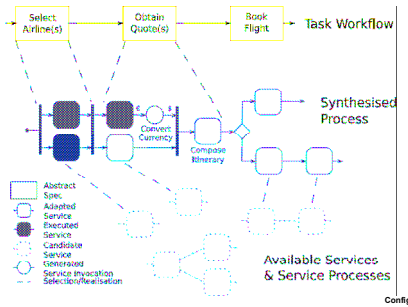
Application 2 – Service Composition

- Main goal: establishing dynamic interaction between applications
- Based on existing work on service composition
  - [Kleiner, Henocque CWS 07] port-based matching (IO), no preconditions or effects (PE)
  - [Thiagarajan, Stumptner ECOWS 07] explicit reasoning on PE's in OCL, 2 level compositoin process

Basic assumptions

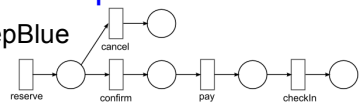
- Services don't exist in a vacuum
  - Composition in the context of an appl. *process* on both sides
  - User prespecifies control flow between tasks that will be tied to abstract services
  - Specification of functional and other requirements for service usage
  - Global requirements (e.g., maximum cost)
- The environment is unreliable
  - Explicit inclusion of control tasks
  - To query environment and reconfigure task execution if necessary
- Not dealing with structural heterogeneity

The Idea

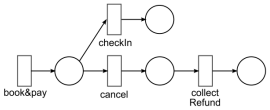


Airline booking service processes

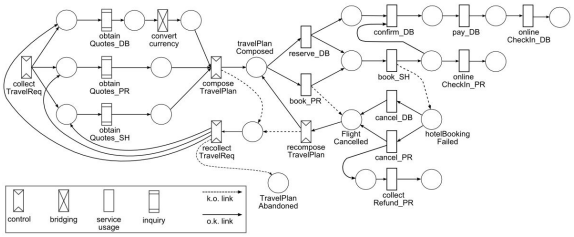
- DeepBlue



- PurpleRed



Detailed Task Workflow - Business Trip



Comparison

- Parts of the model
  - Process model
  - Business objects
  - State and execution schedule
  - A way to conduct composition
- How does it fit the scheme?
  - OO process metamodel
  - Constraint-based rematching given a different pool of component services

Other approaches

- Consistency restoration after user interaction [Amilhastre, Fargier AIJ 02]
  - Compilation of possible choice paths
  - Find a set of actions to restore consistency
  - Static model assumed (Renault example)
- Case-based Reasoning e.g., [Watson, Gardingen 99]
  - Case retrieval (relaxation and matching)
  - Case adaptation

Case adaptation in CBR

- Lots of methods
  - Substitution
    - Reinstantiation, Parameter adjustment, local search, case-based substitution
  - Transformation
    - Commonsense transformations
    - Model-guided repair based on causal relations and functional reasoning [Goel, Stroulia 96]
      - Detailed repair plans based on specific functional differences

Summary

- Overview over reconfiguration approaches
- Action-based vs consistency based approaches
- Some current sample problems

Overview

- Web Services
- Composition Support Schemes
- Semantic Matching Schemes
- Matching as Configuration
- Useful extensions

Web Services for BPI

- Starting point: BPI (Conceptual Modeling)
  - taking complementary (potentially interacting) processes and matching individual operations (not necessarily 1:1)
  - Original intent: Design time/interactive
  - What we are looking at:
    - appropriate semantics for the language
    - Not so much runtime
    - Base models: Petri nets, statecharts, contracts
- Use the WS standards hierarchy? HTTP, SOAP, WSDL (⇝)
  - Encapsulating parts of business logic
  - Hide heterogeneity
  - Standardized & flexible communication

Levels of Composition

- Manual/hardwired choreography
- Composition
  - Dedicated languages for the latter
    - UML EDOC
    - XPDL
    - WS languages (e.g., BPEL4WS)
    - ebXML
- Matching
- Automated Composition

Configuration Overview

- Composition of systems from sets of components

Components/ Assembly	fixed	Parameter-ised	Set of types (catalog)
fixed	Verification	Parametric design	
skeleton (to flesh out)	Assignment		
free	Layout, scheduling		Full config design

[Wielinga, Schreiber] IEEE IS 98 Special issue

Other special issues: AI EDAM 98, AI EDAM 03

- Types of constraints: local, incremental, global (func.)

Configuration Overview

- Composition of systems from sets of components

Components/ Assembly	fixed	Parameter-ised	Set of types (catalog)
fixed	Verification	Parametric design	
skeleton (to flesh out)	Assignment		
free	Layout, scheduling		Full config design

… Planning …

Configuration methods

- KE community: "knowledge-rich methods"
  - e.g., SISYPHUS project (VT elevator example, 89)
  - Typically "propose-revise" or some mod
  - Requires detailed specifications of "fix" strategy
- General configuration community
  - General methods focused on efficient description
  - CSP, FOL based, generic problem solvers
  - Standard ontologies [Soininen et al. , Felfernig et al. 03]: Components, ports (for connections), resource specifications (aggregate functions)
  - Complex constraints over large scale assemblies (multiple instantiation)

Support for composability checking

- Methods generally from DB or conceptual modelling community
- Syntactical correspondence of properties [Yang/Papazoglou CAiSE 02]
- Matching aggregation hierarchy [Fileto et al. VLDBJ 02]
- "Ontology-based matching": Extensive list of properties (e.g., purpose, quality for bindings, parameters, function) that are matched textually (with synonyms) [Medjahed et al. VLDBJ 03]
- All have special purpose algorithms and fixed set of properties

OWL-S based support

- Mainly service *matching*
- Only control patterns within a single service
- Use standard DL subsumption reasoner
  [Paolucci et al. 02] [Li, Horrocks WWW 03]
  - Exact: exact match or request **is-a** offer
  - PlugIn: request $\sqsubseteq$ offer
  - Subsumes: offer $\sqsupset$ request
  - Failure: absence of subsumption

Preference Ordering

Li/Horrocks Example

- items are provided by an Actor with name "Georgia";
  items are PCs and the memory size is at least 128 Mb;
  the quantity of PCs being bought will be less than 200;
  the unit price is more than 700;
  the seller must have a creditLevel greater than 5;
  goods must be delivered before the 15/09/2002;
  goods must be delivered in Bristol.
- **Advert1** = ServiceProfile ⊓ (Sales ⊓
  ∀providedBy:(Actor ⊓ ∀ hasName:Georgia) ⊓
  ∀ requestedBy:(Actor $\geq_5$ hasCreditLevel) ⊓
  ∀ item:(PC ⊓ $\geq_{128}$ memorySize) ⊓
  $\geq_{700}$ hasUnitPrice ⊓
  $\leq_{200}$ hasQuantity ⊓
  ∀ delivery:(Delivery ⊓
  $\leq_{20020915}$ date ⊓ ∀location.Bristol))
- **Query** = ServiceProfile ⊓ (Sales ⊓
  ∀ providedBy:(Actor $\geq_5$ hasCreditLevel) ⊓
  ∀ item:(PC ⊓ ∀ hasProcessor:Pentium4
  $\leq_{700}$ hasUnitPrice))

A different interpretation

- Consider functionality as key components (function is type of component)
- A request is a constraint (user requirement) on a service
- Matches expressed through consistency ➔ service matching defined as a configuration problem
- Given: Solution CONF, service profile S, request R
  - S ∈ CLANG, R ∈ CLANG [Felfernig et al.03]
  - DD ∪ SRS ∪ CONF satisfiable
  - Then a = @A ∈ CONF is a matching service if
    {R' subclass R, r = @R} and r.sv=a ∈ CONF}

Example

- **Service** [Li, Horrocks 03]

```
class Seller subclass Actor . . .
      seller1 = @Seller
      name = "Georgia";
      end;
class Sale1 subclass @CompSale
      seller = seller1;
      item = "PC";
      memory =128;
      processor = Set{Pentium3, Pentium4, Athlon...}
      deliveryLocation = "Bristol";
inv quantity <= 200 end;
inv unitPrice >= 700 end;
inv seller.creditLevel > 5 end;
inv deliveryDate < 15/09/2000 end;
```

Example (Cont.)

- Request
  - **class** CompOrderDescription **subclass** ServiceRequest **end**;
    saleOrder = @SaleOrderDescription
    sv = @CompSaleAd;
    sv.item = "PC";
    sv.processor = "Pentium4;
    inv (sv.seller = @Actor) **and** (sv.seller.creditLevel > 5)
    inv sv.unitPrice **end**;
    end;
- no requirements beyond normal (multi-component) configuration reasoner
  - Subsumption only through type hierarchy (key components)
  - Constraint/logic/resource based

Other approaches

- [Constantinescu, Faltings, Binder WIS 04]
  - All ports specified through types (intervals)
  - Complete/partial: consistency-based match via resource constraints
  - No other constraints (all semantics compiled into port names), e.g., basePrice ➔priceMinusRebate ➔ priceWithTax1 ➔ priceWithTax2
  - No side properties (e.g., qos)
- [van Harmelen, ten Teije, Wielinga ECAI 04]
  - Classical propose-revise approach
  - Main argument: no planning possible because pre/post specs won't be available

That's just a start

- Optimization
- Open CSP [Faltings et al.03]
- Exact boundary to planning problems
  - [Ponnekanti/Fox 02] [Hendler, Nau et al.03] [Sheshagiri, desJardins, Finin 03]
  - Precondition/effect specifications
- Prespecified control flow
  - Would be "Skeleton" style configuration except for control structures
- Requirements Matching: "semantic" UDDI e.g., [Mandell, McIlraith 03]

What's the goal?

- Languages useful enough for commercial applications
- Effective GP reasoning mechanisms for them
- Re-evaluation of overlaps between areas related of old (planning and configuration)
- Conceptual Modeling community (integration based on semantic behaviour specifications)

Summary

- WS Matching as a configuration problem
- Existing formalisms easily adapted (not very surprising, see SW config modeling)
- Re-use of standard algorithms
- A lot of past work can be adapted
- Utility depends on criteria