ALPEN-ADRIA
UNIVERSITÄT
KLAGENFURT

Constraint-based personalized bundling of products and services

Markus Zanker, Markus Aschinger
and Markus Jessenitschnig
University Klagenfurt

Contents

- Approach
- Motivating example
- Process steps
- Evaluation
- Conclusions

Motivation

- Configurators perform a special type of design activity by synthesizing an artifact from a set of pre-defined components under observance of domain restrictions.
- Recommender systems (RS) suggest items to users according to their estimated preferences and their needs situation.
- E-tourism scenario: Guests should receive **personalized** service bundles, e.g. leisure activities, restaurants, sights or shopping opportunities.
- However, RS are typically not capable of recommending consistent groupings, bundles or configurations in a wider sense to users while configurators are not capable of personalizing their results (based on stochastic methods)!

Approach 1/3

- Constraint-based approach for synthesizing configuration and recommendation technology

- Extension of the configuration approach through
  - pre-filtering of the problem space by the use of different recommenders (considering implicit preference information)
  - Composition problem modeled as a CSP

- For each component a RS computes a ranked list of recommendations that form its domain

- Configurator exploits domain knowledge and consistency conditions defined by constraints

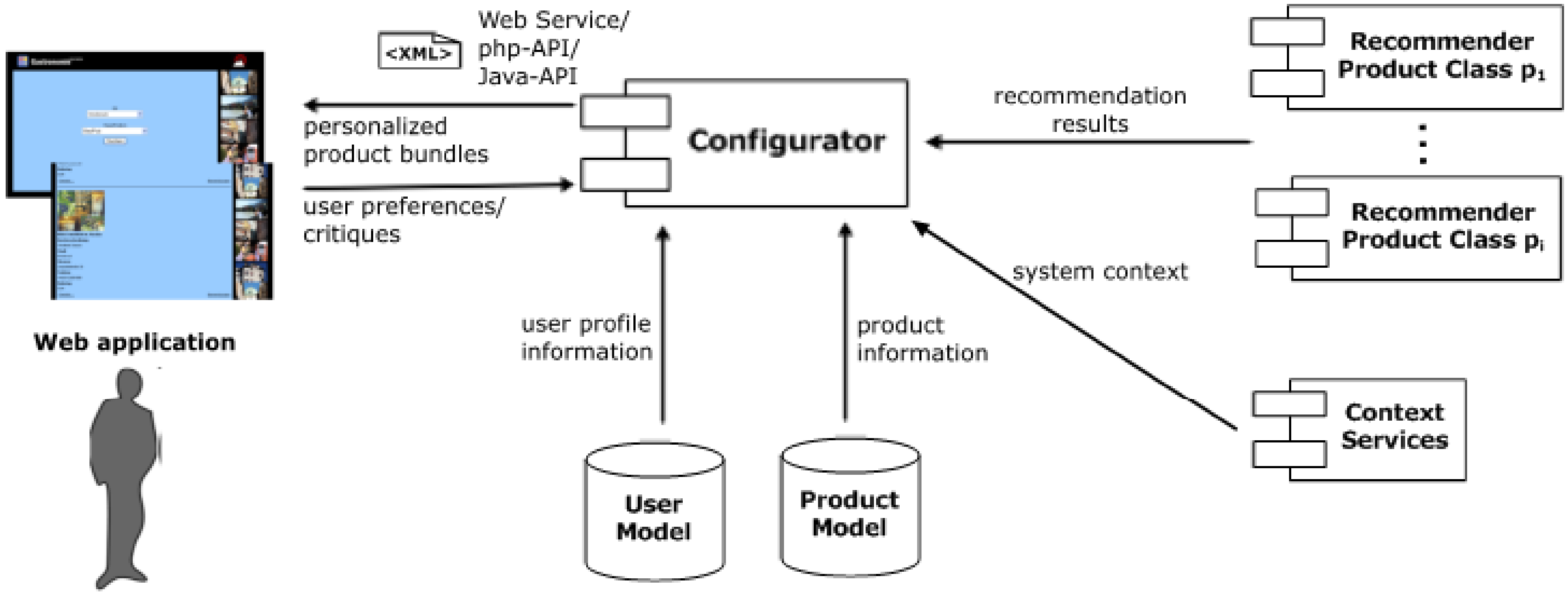
Approach 2 / 3

- extending the configuration approach through
  - pre-filtering of the problem space by the use of different recommenders (e.g. collaborative filtering)
  - consideration of implicit preference information

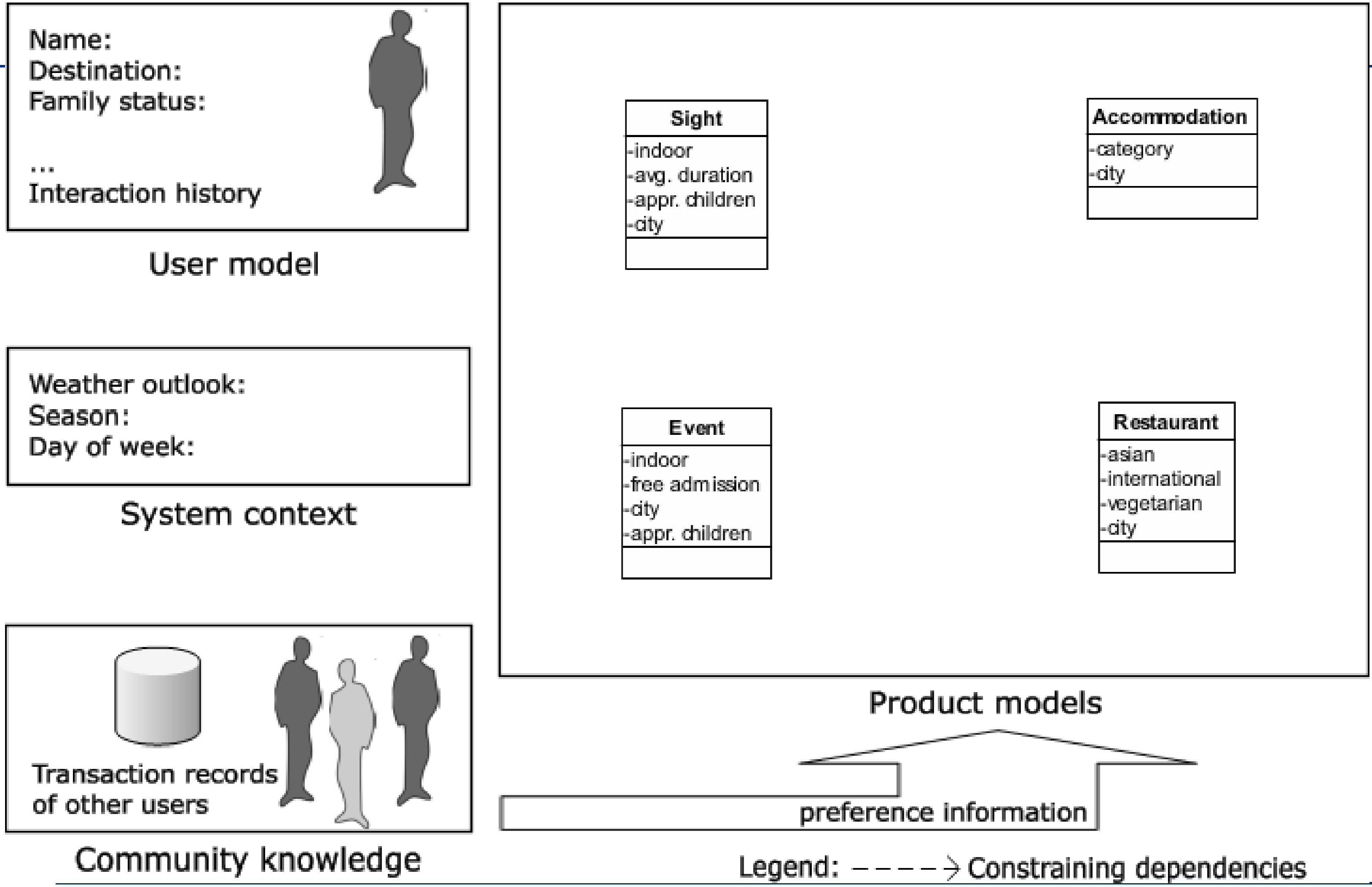
Collaborative filtering with single rating table

	Absolut Inn	La Cabana	Solo-vina	Kunst-raum I.	Galerie Taxisp.	Alpen-zoo	User similarity
John	1	1	●	1		●	Recommendation
Jim	1	1	1			1	0.58
Helen			1	1	1		1/3
Eve						1	0

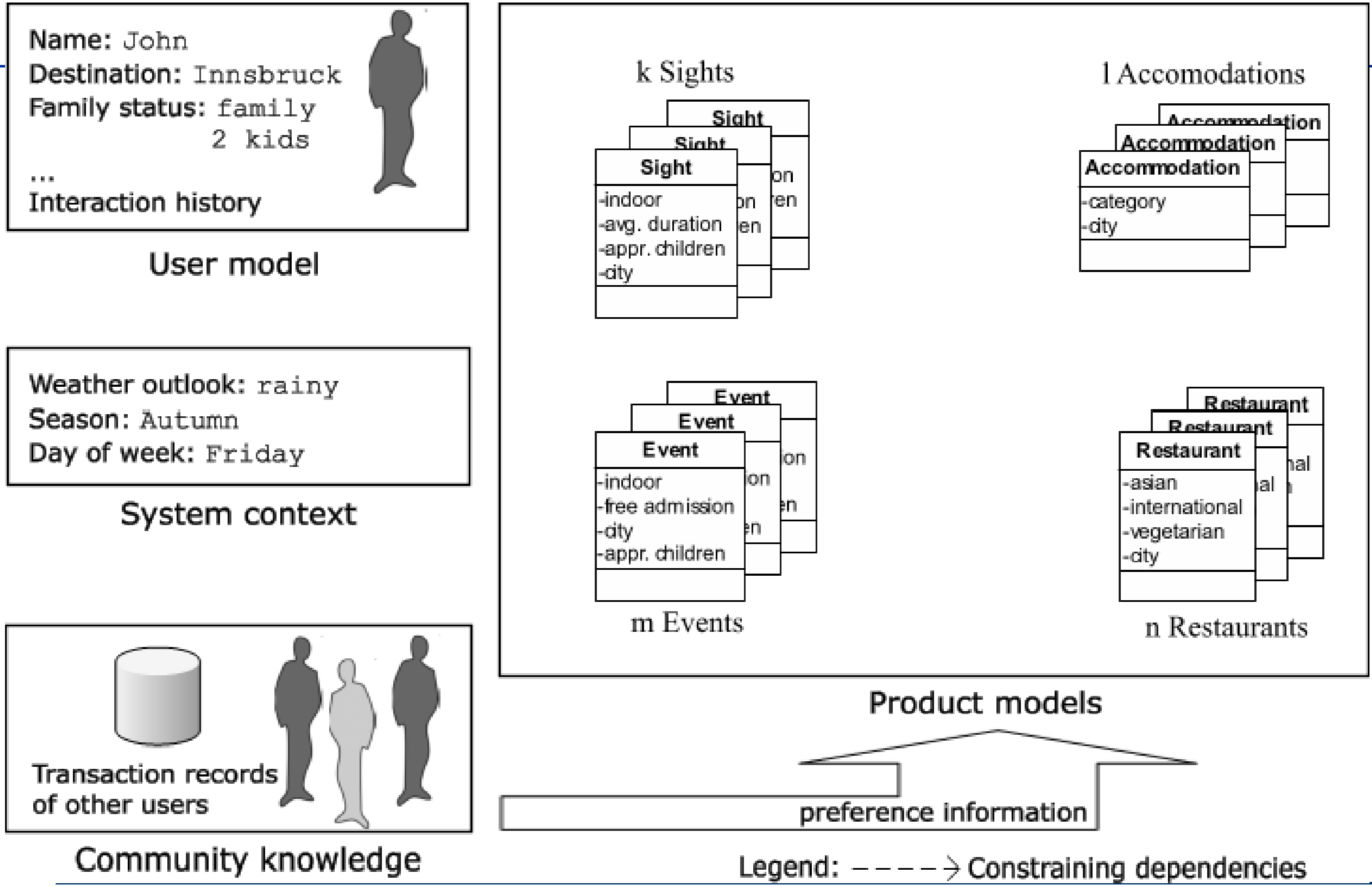
Approach 3 / 3


<XML>
Web Service/
php-API/
Java-API
personalized
product bundles
user preferences/
critiques
Configurator
recommendation
results
Recommender
Product Class p_1
Recommender
Product Class p_i
system context
user profile
information
product
information
Context
Services
User
Model
Product
Model
Web application

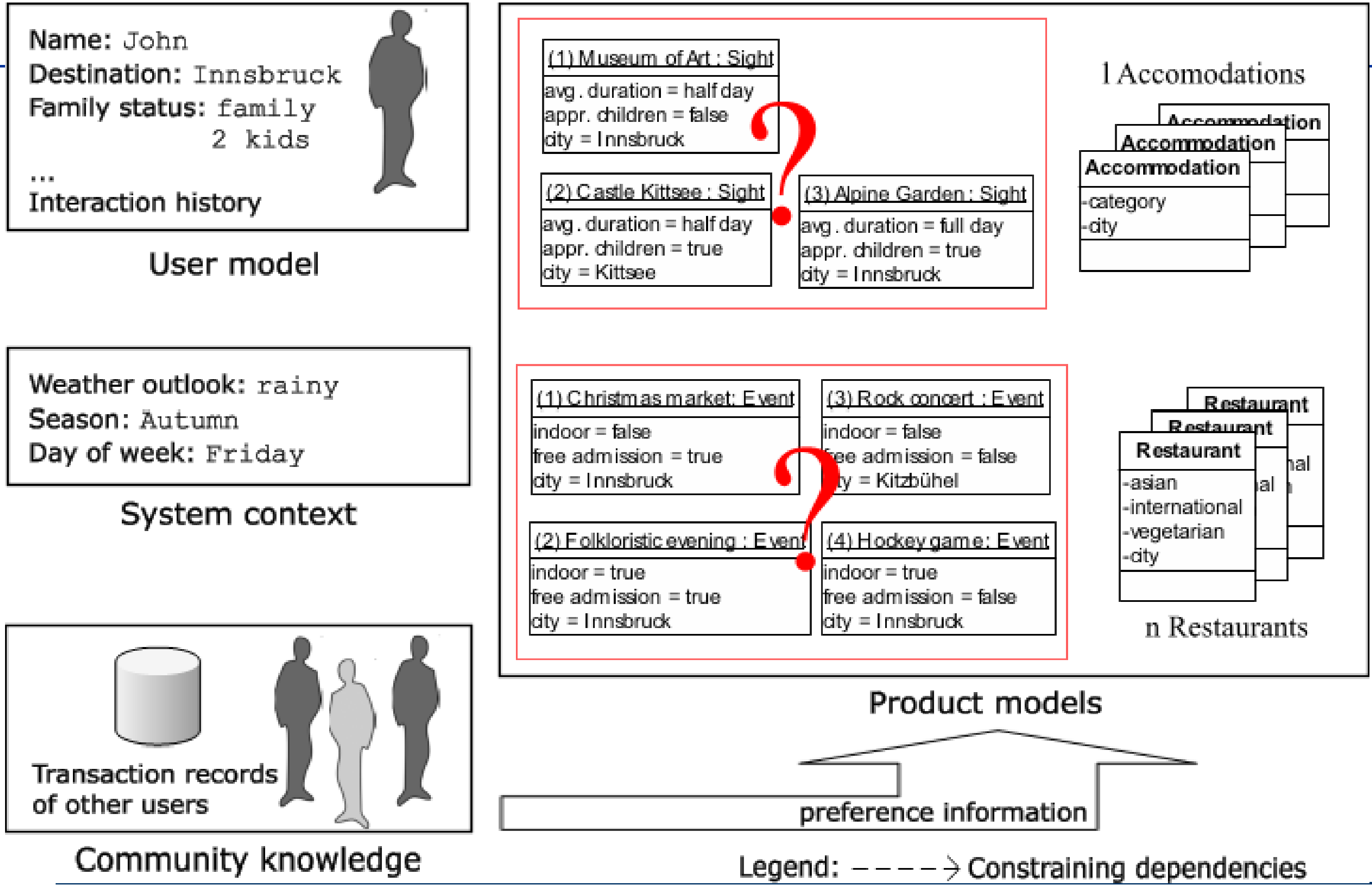
Motivating Example (1)



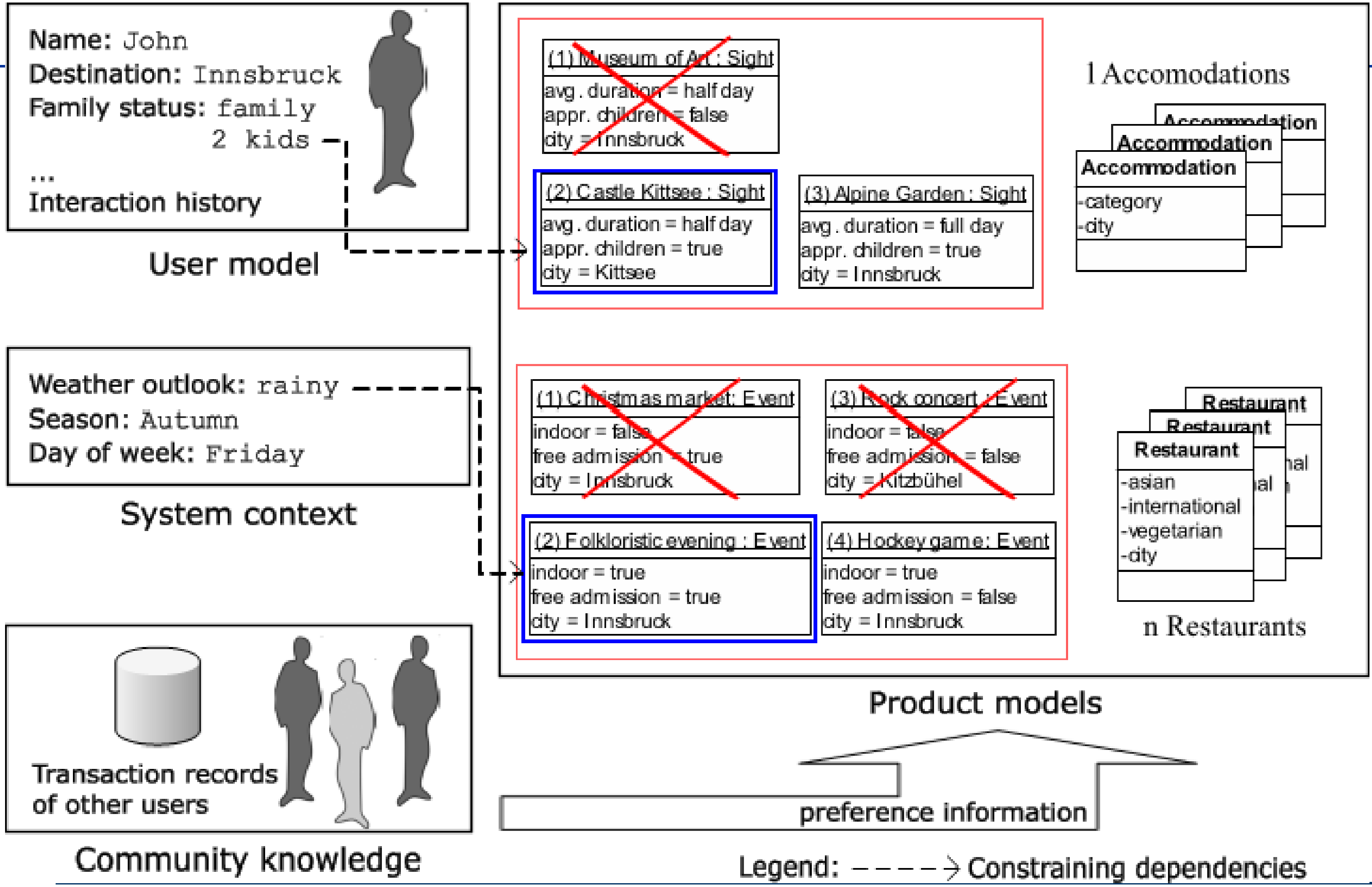
Motivating Example (2)



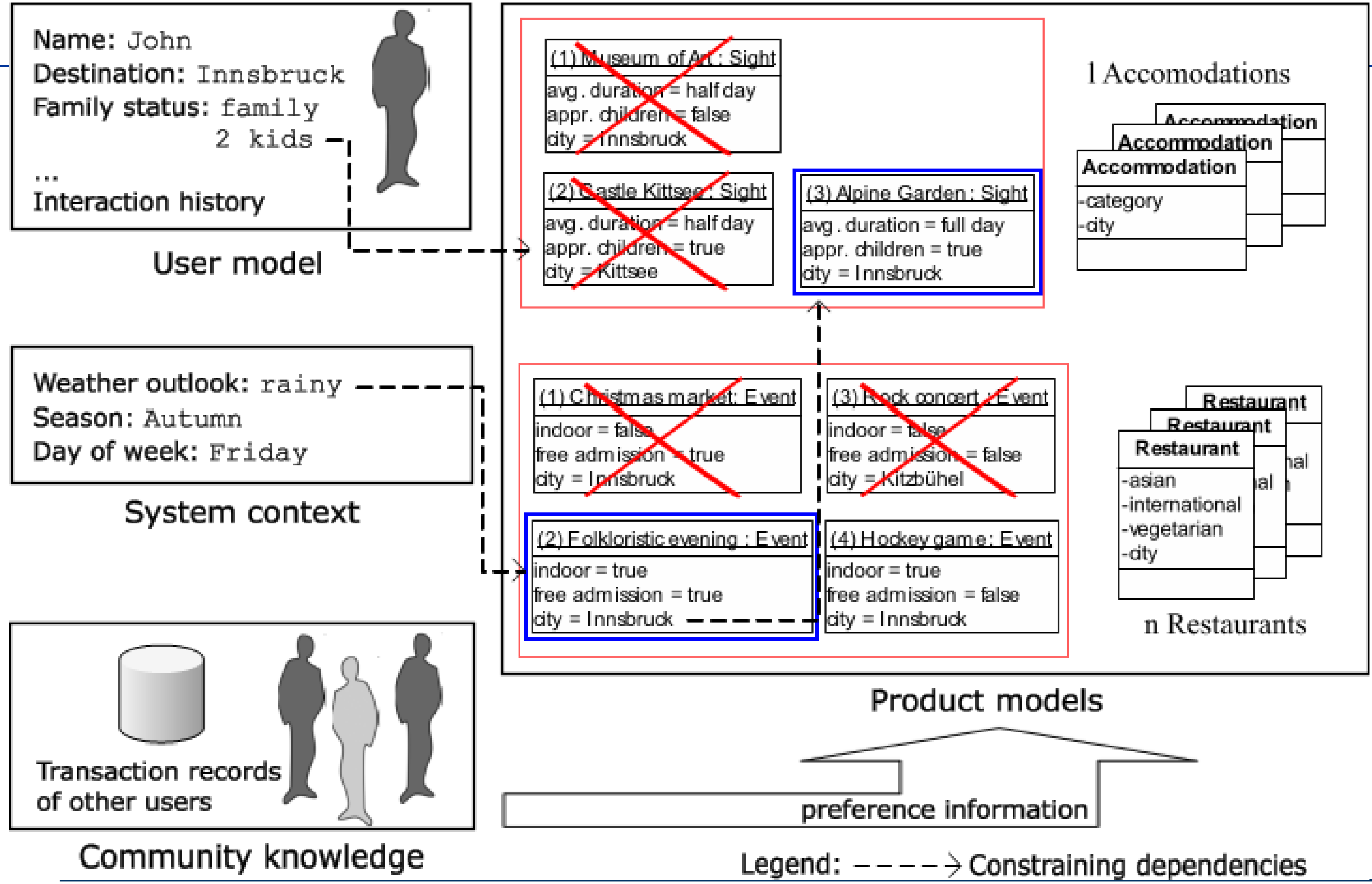
Motivating Example (3)



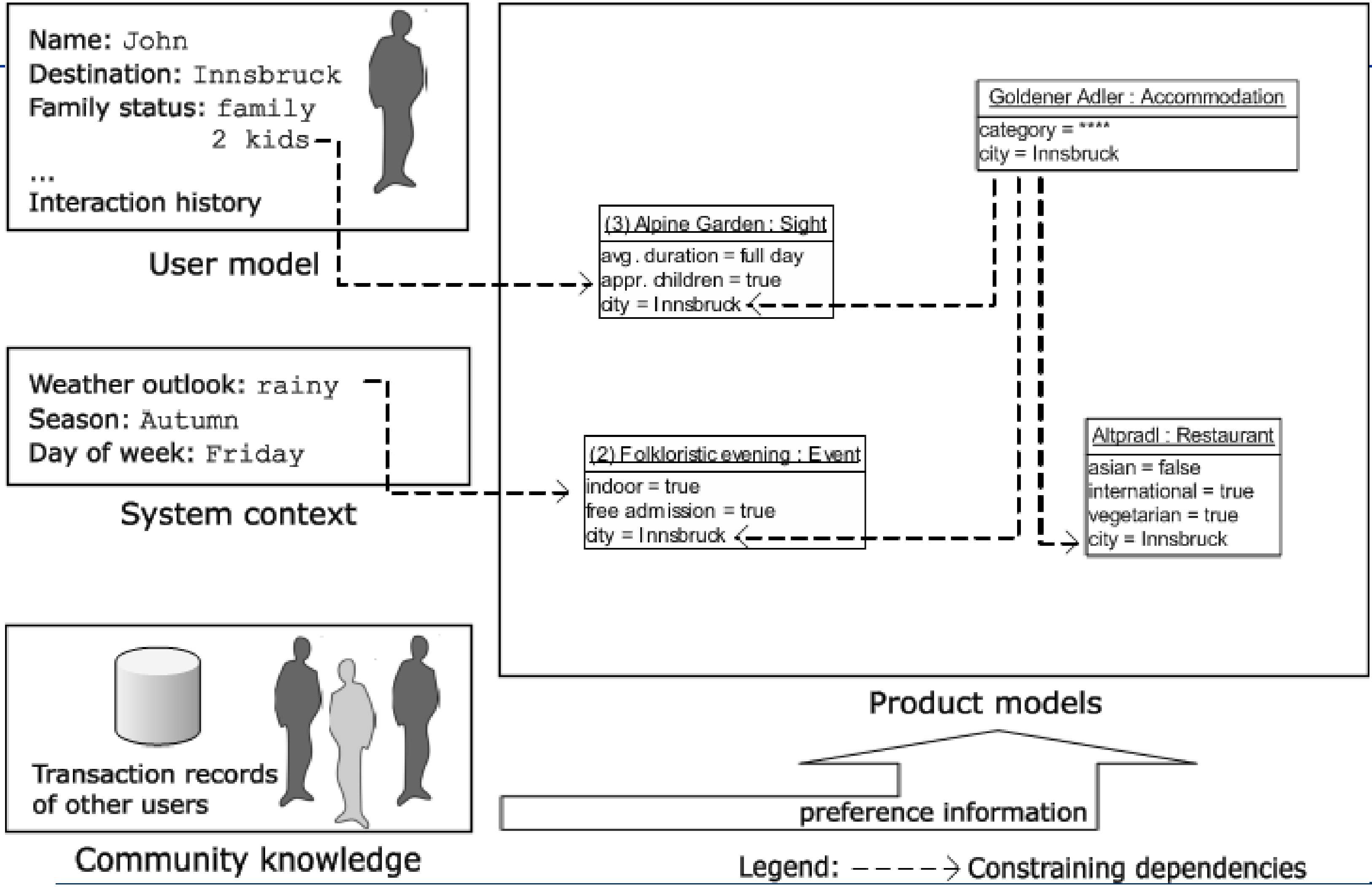
Motivating Example (4)



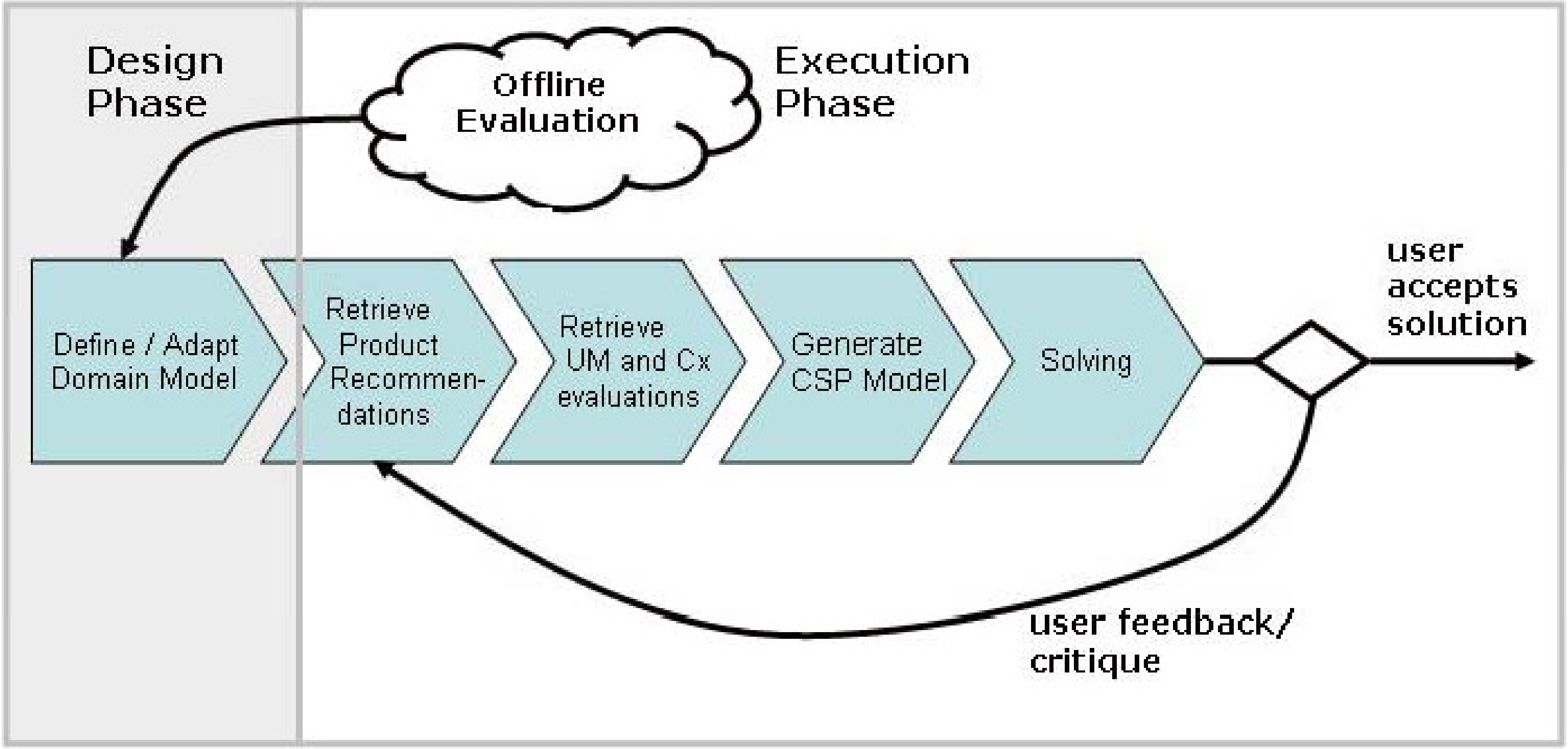
Motivating Example (5)



Motivating Example (6)



Process steps



Obtaining domain data

- Retrieve product instances
  - retrieve a ranked list of instances for each product category
  - request the corresponding product characteristics from the product model repository
  - standardized calls by the use of a generic recommender API
  - currently: hard-wired allocation between a recommender and a product category
  - future outlook: intelligent selection strategy

- Retrieve UM and context information
  - arbitrary queries over the user profile for the UM variables
  - calculation of context variable values via external functions

CSP Generation

- Variables
  - create all variables in $X_{UM} \cup X_{Cx}$ and assign them their respective evaluations
  - create an index variable p.idx for all product categories $p \in P$ with the domain $p.d_{idx} = \{1, ..., p_n\}$, where 1 denotes the highest ranked product instance and p_n the lowest ranked one
  - create all product properties in X_{PM} and assign them domains where all $p[i].x \in p.d_x$
- Constraints
  - insert all domain constraints from $C_{hard} \cup C_{soft}$
  - add the explicit user constraints for the actual session
  - secure the consistency of components by the use of integrity constraints in the form $p.idx = i \rightarrow p.x = p[i].x$
- Optimization
  - create a penalty variable c.pen for each soft constraint $c \in C_{soft}$
  - create the resource variables and the corresponding optimization constraints

CSP Solving

- **Goal:** find an assignment to all variables in the CSP model that does not violate any hard constraint and optimizes the bundles considering
  - the ranking of objects for each product category and
  - the fulfillment of the soft constraints
- trade-off decisions
  - relax a soft constraint or choose a lower-ranked alternative product instance?
- different solving strategies
  - different semantics of **next solution**: no / only some components may overlap in two bundles / configurations
  - 1-different / all-different / (n-different)

Optimization model

$$\textbf{min} \quad WF * RV_{PROD} + (10 - WF) * RV_{SOFT}$$

subject to

$$\sum_{i=1}^{m} prio_i * \frac{OV_i}{\# DO_i} * \frac{100}{m} = RV_{PROD}$$

$$\sum_{j=1}^{n} SC_j * \left[\frac{penalty_j * 100}{n} \right] = RV_{SOFT}$$

where

$m \ldots \#$ of product categories

$n \ldots \#$ of soft constraints

$\# DO_x \ldots$ received instances for product category x

$prio_x \in [1, \ldots, 100] \ldots$ priority for product category x

$penalty_x \in [1, \ldots, 100] \ldots$ penalty for soft constraint x

$OV_x \in [1, \ldots, \# DO_x] \ldots$ rank of product instance x

$SC_x \in [0, 1] \ldots$ fulfillment of soft constraint x

Session- & Solution-Management

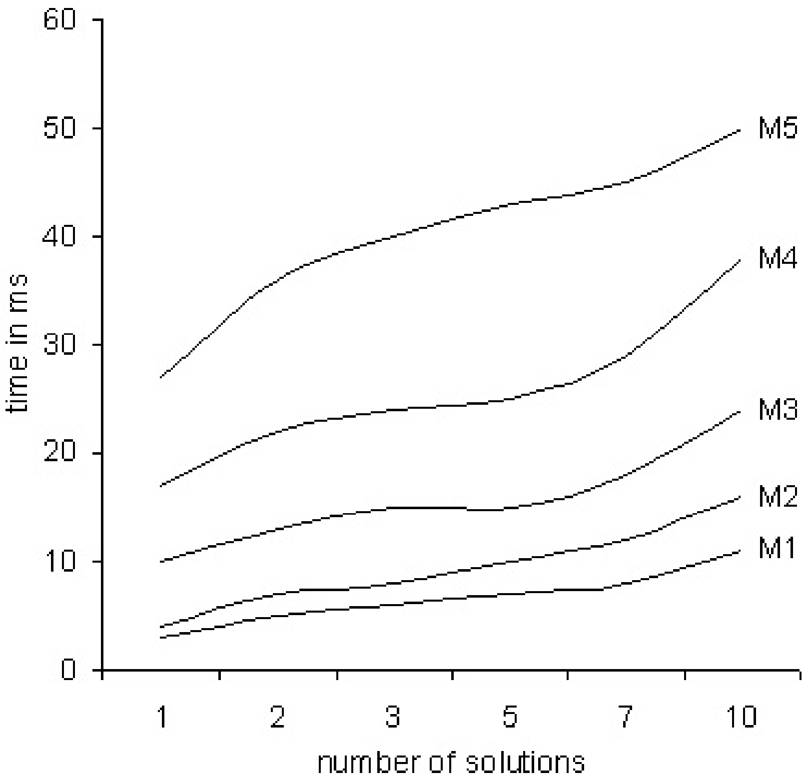
- interactivity between the system and the user during exploration of the search space
- long-lasting sessions
  - configuration sessions are stored in the user model and can be resumed
- further usage of partial solutions
- add / modify / delete constraints and preferences during each interaction step

Evaluation

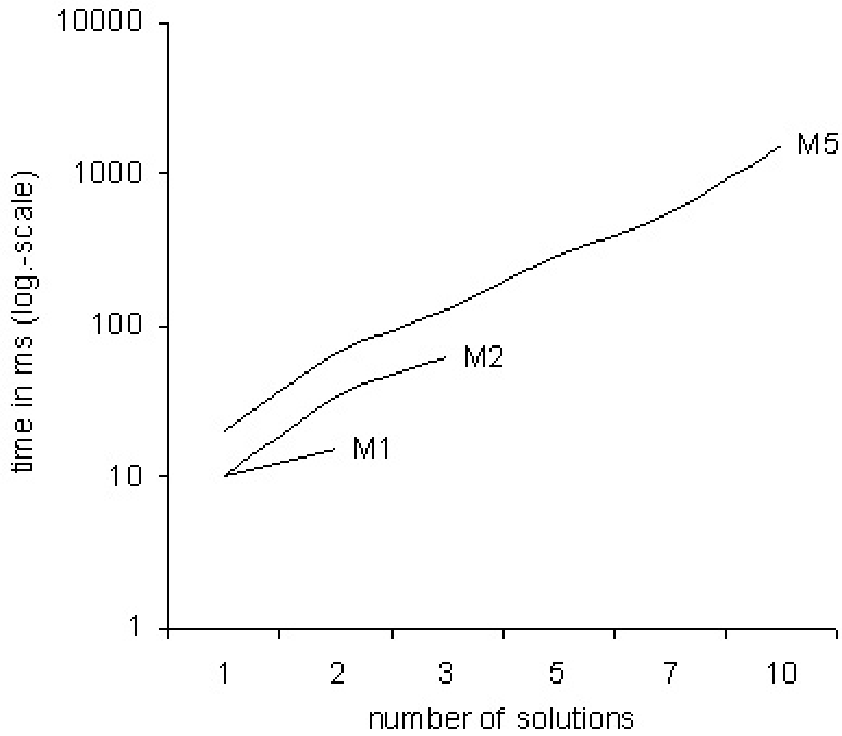
- Dataset from E-tourism platform *innsbruck.mobile*
  - Product base with 3000 items
  - Interaction log with 4195 entries from 884 users
- Example scenario
  - 5 product classes, 30 product properties
  - 23 domain constraints (13 hard and 10 soft)

Model	Number of Recommendations	Number of Vars	Average Domain Size	Number of Constraints	Generation time in ms
M1	5	58	7,45	206	10
M2	10	58	8,73	374	20
M3	30	58	13,55	1010	60
M4	50	58	16,5	1355	95
M5	100	58	23,23	2093	135

Evaluation



1-different



all-different

Conclusions

- Novel strategy to personalize configuration results on product bundles using recommenders
- Solving of standard product bundling tasks in online sales situations showed acceptable computation times
- Future work
  - Evaluation w.r.t. user satisfaction
  - Experiment with different optimization functions
  - Handling of over-constrained problems
    - E.g. Dynamic domain extension

Thank you for your attention!

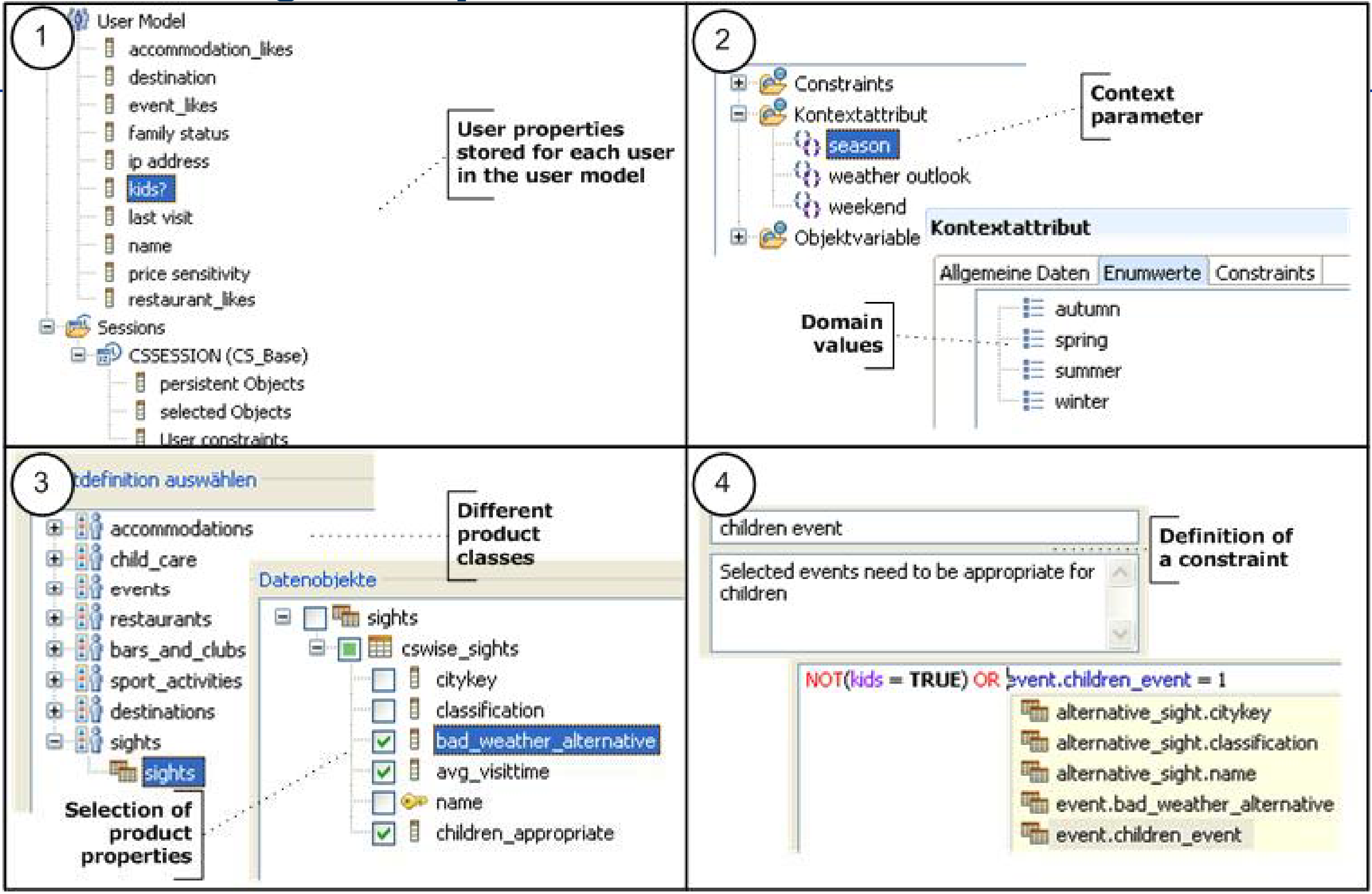
Questions?

CSP-Model

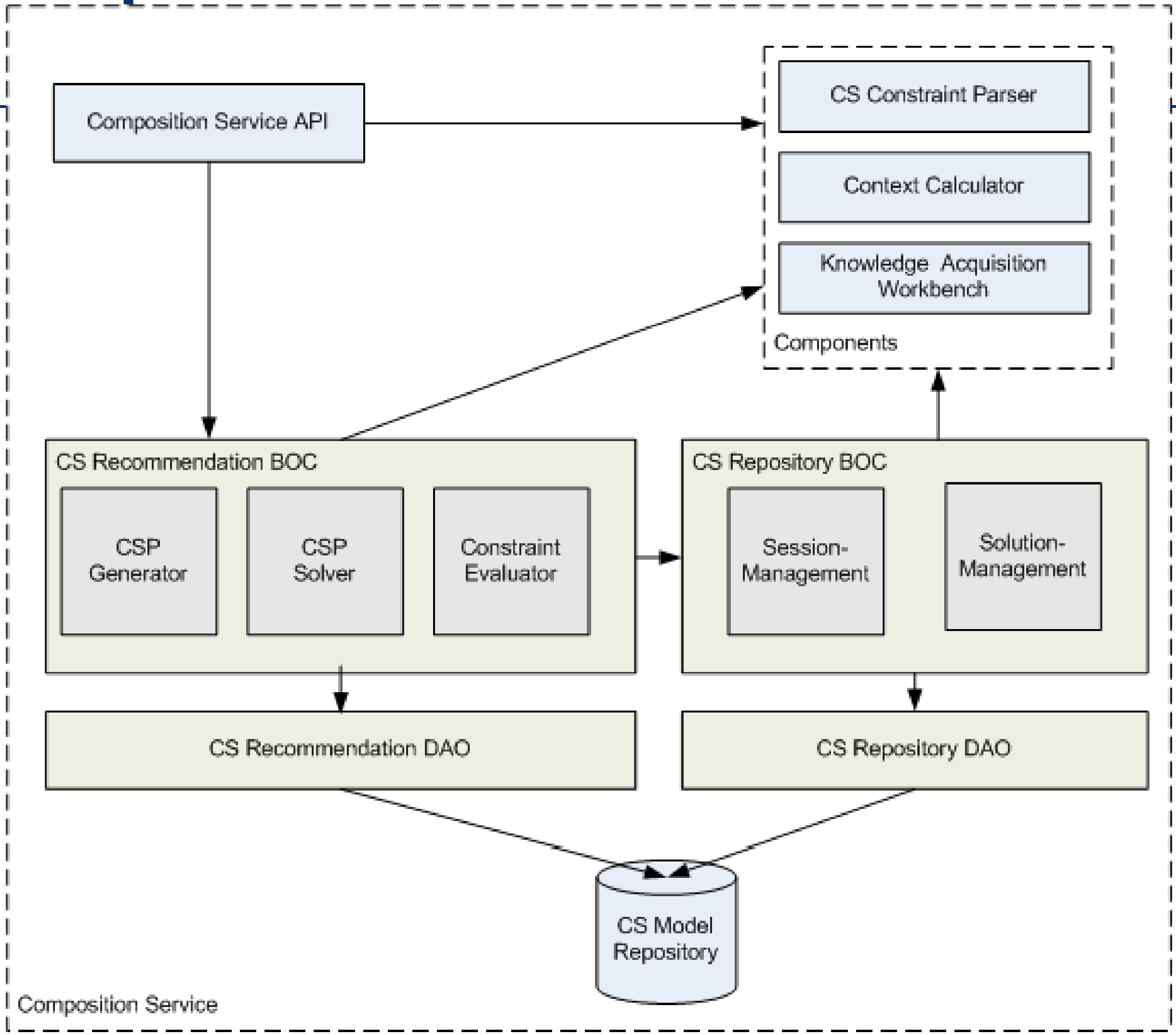
The CSP-model consists of a tuple $\langle X_{\{P, UM, Cx, PM, RV\}}, D_{\{P, PM\}}, C_{\{hard, soft\}}\rangle$, where:

- $X_P = \{x_1, ..., x_i\}$ a set of product categories,
- $X_{UM} = \{x_1, ..., x_j\}$ a set of variables representing properties of the user model,
- $X_{Cx} = \{x_1, ..., x_k\}$ a set of variables modeling the system context,
- $X_{PM} = \{p_1.x_1, ..., p_1.x_m, ..., p_i.x_1, ..., p_i.x_n\}$ a set of variables modeling product properties,
- X_{RV} a set of resource variables for the optimization,
- $D_P = \{d_1, ..., d_i\}$ a set of corresponding domains for product categories,
- $D_{PM} = \{p_1.d_1, ..., p_1.d_m, ..., p_i.d_1, ..., p_i.d_n\}$ a set of corresponding domains for product properties,
- $C_{hard} = \{c_1, ..., c_p\}$ a set of hard constraints on variables in $X = X_{UM} \cup X_{Cx} \cup X_{PM}$,
- $C_{soft} = \{c_1, ..., c_q\}$ a set of soft constraints on variables in X,
- *weight(x_i)* the relative weight of product category x_i in the optimization and
- *pen(c_q)* the penalty value for the soft constraint c_q.

Knowledge Acquisition Framework



Composition Service Architecture



Evaluation

- 1-different

Model	Number of solutions						
	1	2	3	5	7	8	10
M1	3	5	6	7	8	8	11
M2	4	7	8	10	12	14	16
M3	10	13	15	15	18	22	24
M4	17	25	25	25	29	34	39
M5	25	37	38	43	45	48	50

- all-different

Model	Number of solutions						
	1	2	3	5	7	8	10
M1	10	15					
M2	10	33	60				
M3	15	37	65	240			
M4	15	40	90	260	565	1030	
M5	20	65	125	285	550	755	1540

Branch & Bound algorithm

Input: $n \ldots$ *number of desired product bundles*

Output: $solutions \ldots$ *list of solutions in descending order*

$upperBound \leftarrow +\infty$

$solutions \leftarrow array\,[1, \ldots, n]\,of\,integer$

while $\#solutions < n$ **do**

  $solution \leftarrow getNextSolution()$

  insert $solution$ in $solutions$

if $\#solutions < n$ **then**

  return $solutions$

$upperBound \leftarrow$ value of the current n-th solution $- 1$

while *new solution found* **do**

  $solution \leftarrow getNextSolution()$

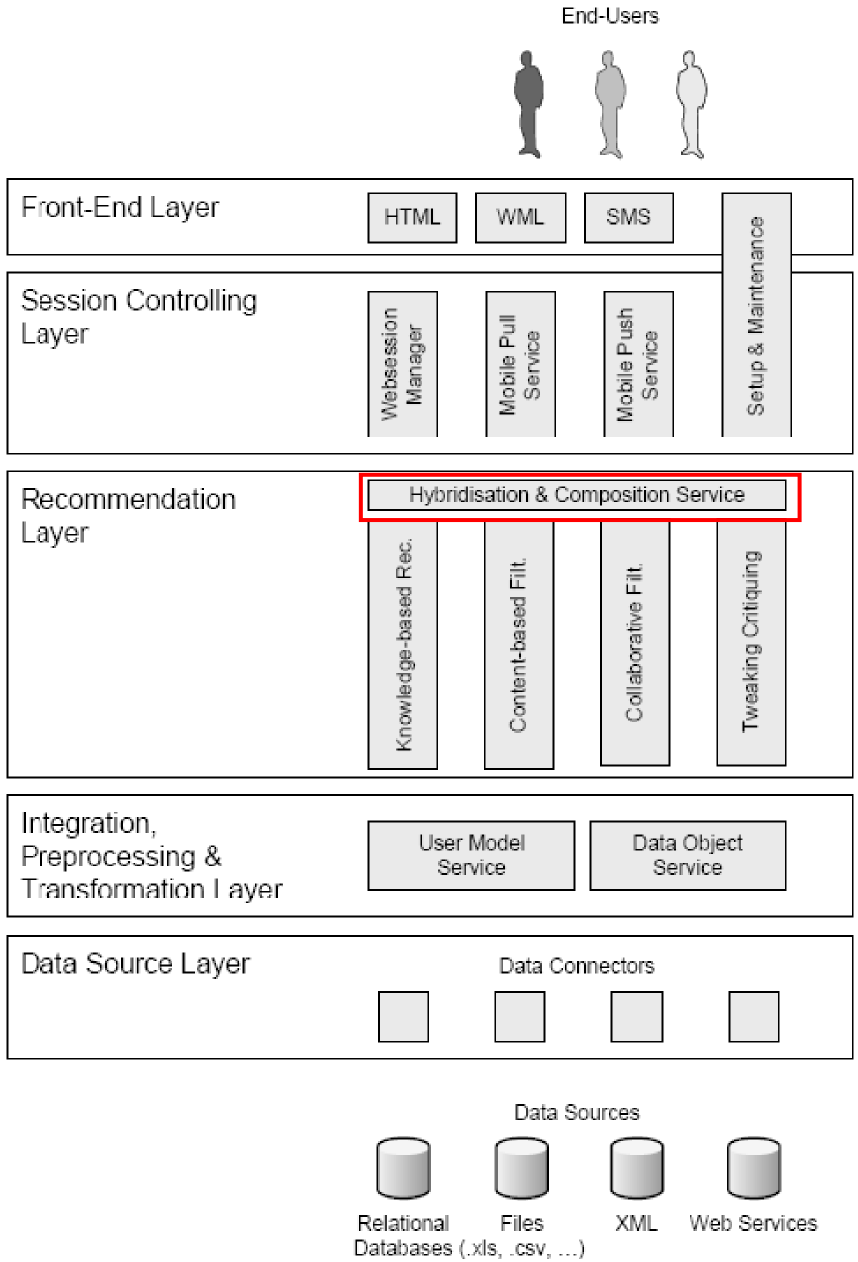
  insert $solution$ in $solutions$

  $upperBound \leftarrow$ value of the current n-th solution $- 1$

return solutions

System architecture

- Front-End Layer
- Session Controlling Layer
  - Push-/Pull-Service
- Recommendation Layer
  - Recommender-Components
  - Hybridisation & Composition Service
- Integration, Preprocessing & Transformation Layer
  - Data Object Service
    - Product data
  - User Model Service
    - User profiles
- Data Source Layer
  - Access on external data sources



CHOCO Constraint Solver

- constraints encapsulate a dedicated filtering algorithm and maintain their own level of consistency
  - arc consistency
  - bound consistency
- event-based propagation engine
- backtracking with depth-first search
- extensions
  - fixing user-defined binary constraints (AC2001)
  - modified Branch & Bound algorithm