intelligent views

Knowledge Management with K-Infinity

Jan Schümmer, intelligent views
Darmstadt, Germany
j.schuemmer@i-views.de

Outline

intelligent views

In this talk, I am going to

- show an application - K-Infinity
- give you a short introduction to the COAST framework
- explain how COAST was used to build K-Infinity
- hopefully encourage you to use COAST in your own applications

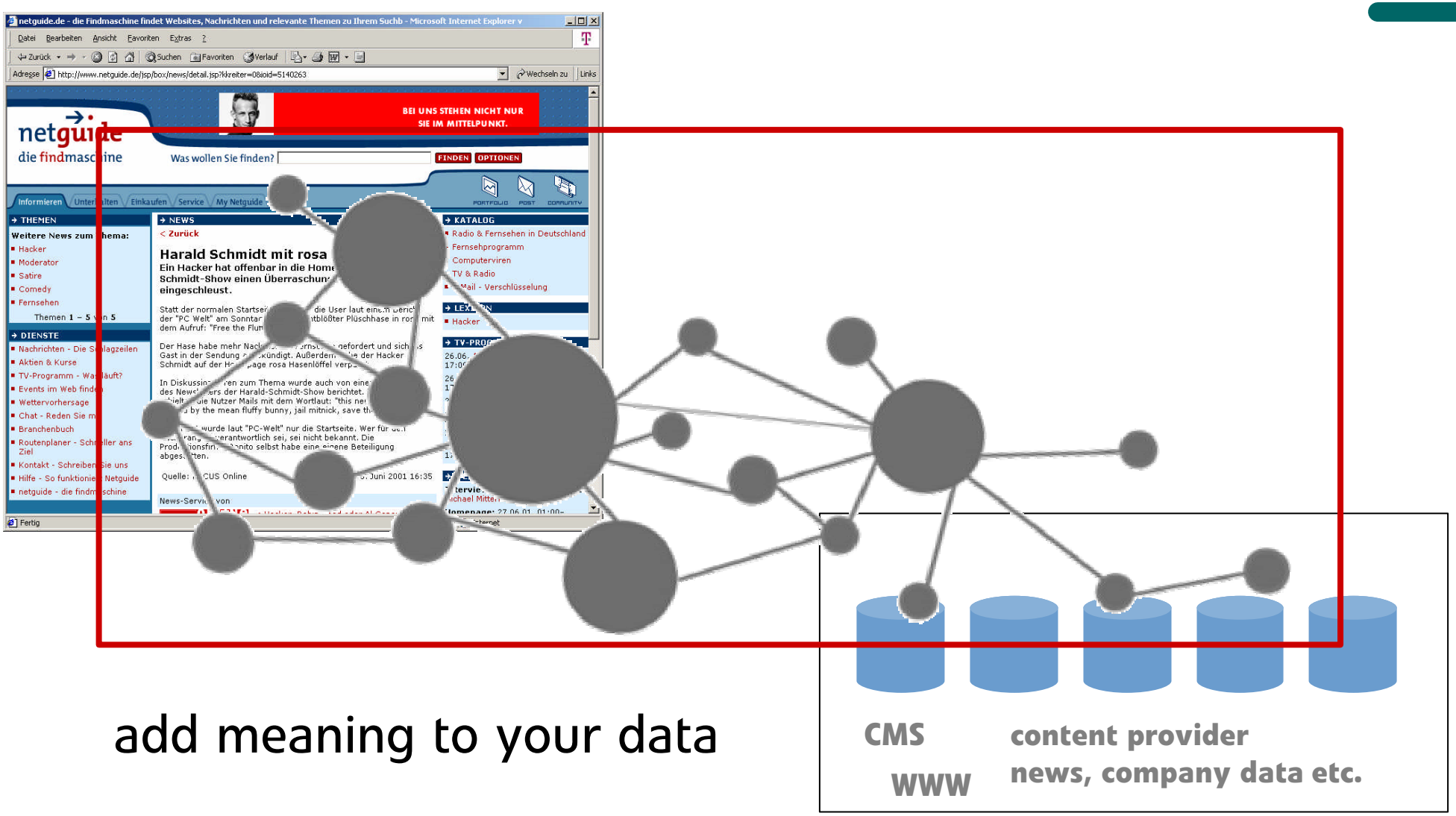
Company Information

intelligent views

- located in Darmstadt, Gemany
- founded in 1997
- spin-off enterprise of GMD – National Research Center for Information Technology, institute IPSI
- about 40 employees

What we are doing

intelligent views



add meaning to your data

Looking for information: WEB portal

intelligent views

Scenario

- You are visiting a classical music news portal to find information about conductors in Bayreuth

Full text query does not help, because

- it only finds strings, not concepts
  - no synonymes
  - no translations
- it produces irrelevant hits
  - homonymes
  - different contexts

A model is needed

intelligent views

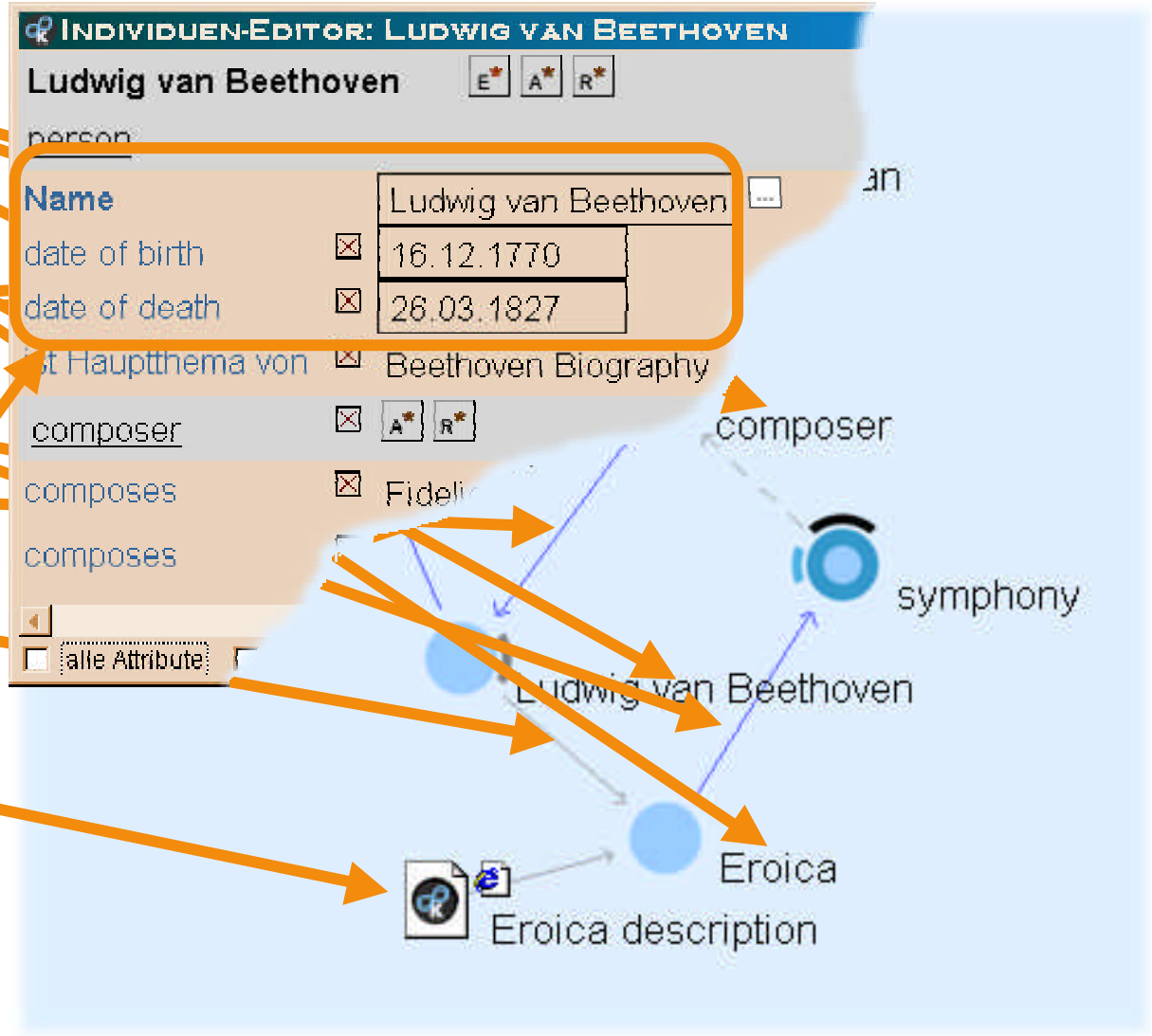
K-Infinity allows to

- express knowledge in terms of an object-oriented model
- connect content to arbitrary entities of this model
- use the knowledge net for
  - semantic search
  - presenting related things next to each other
  - … reasoning

Building blocks

intelligent views

- Concept
- Individual
- Relation
  - inheritance
  - instance of
  - extension
  - user defined relations
- Document
- Attribute



Demonstration

intelligent views

- create individual
- create role concept
- define relation for role
- assign role to individual
- create instance of the newly defined relation

- import document
- connect it to the knowledge net
- show matching environment (related documents)

K-Infinity tool suite

intelligent views

Knowledge Builder

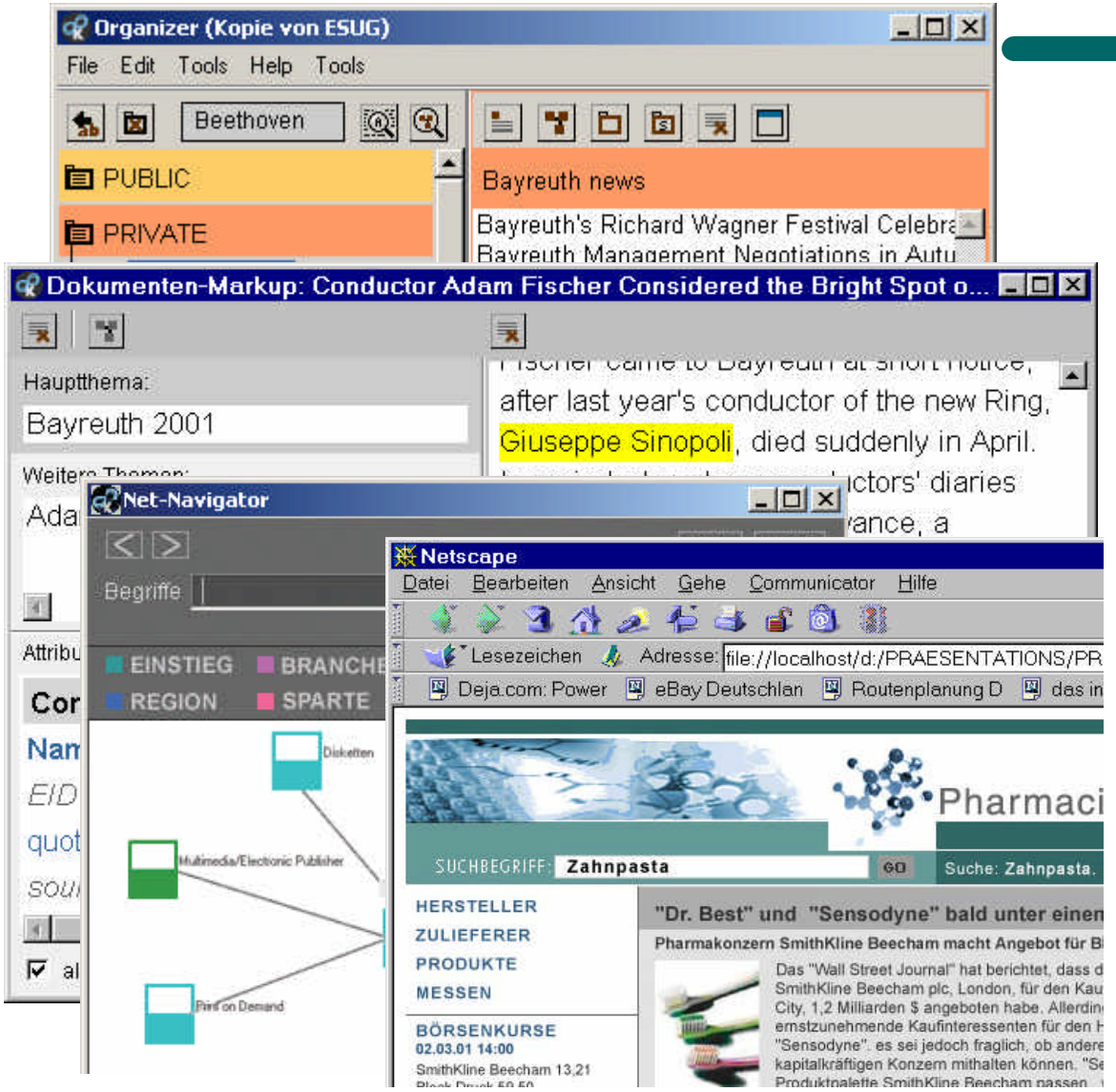
- schema definition

Markup Tool

- link documents to the net

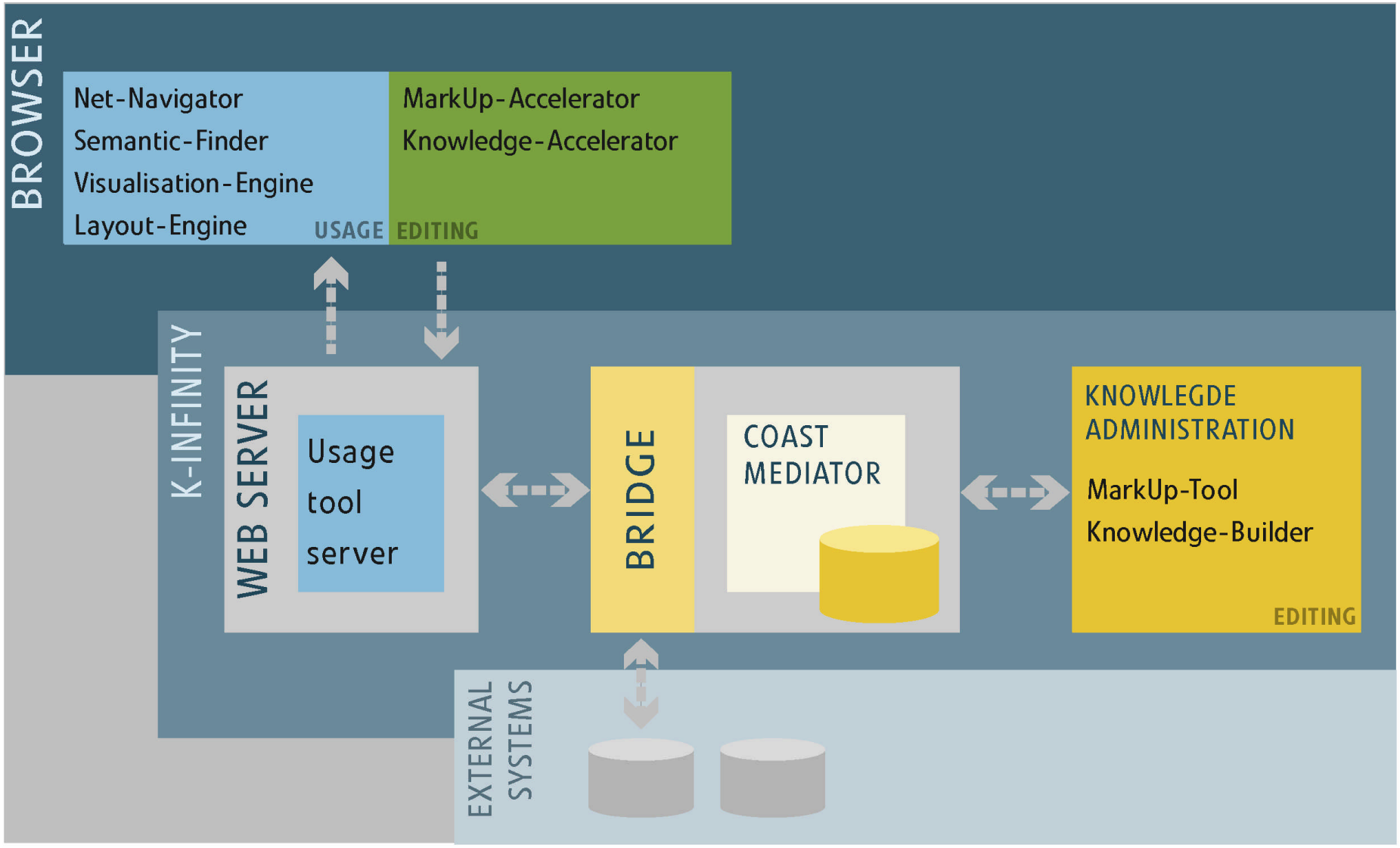
Usage tools

- Net Navigator
- Knowledge Accelerator
- Web presentation engines



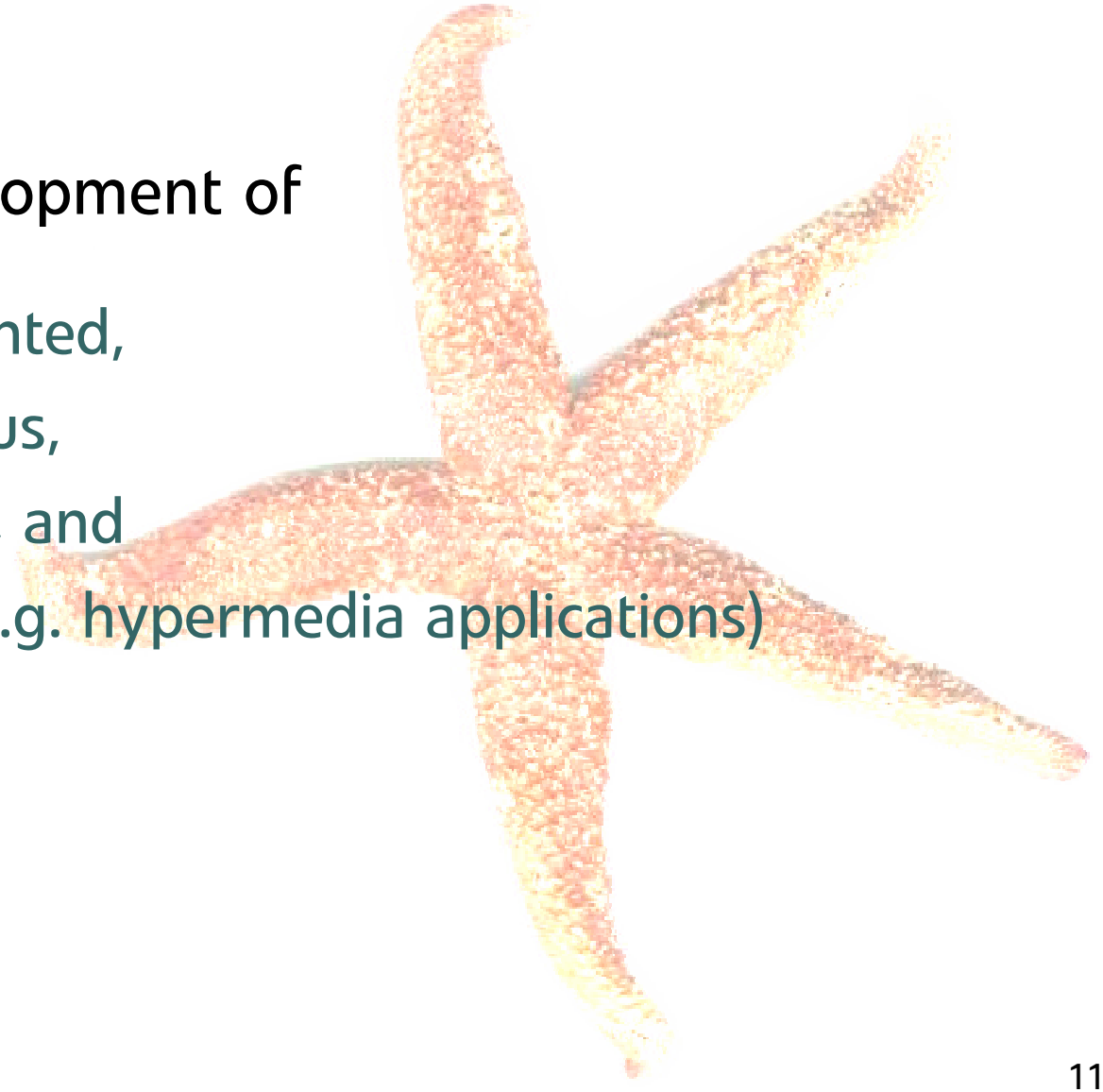
Behind the Scenes

intelligent views



The COAST Framework

intelligent views



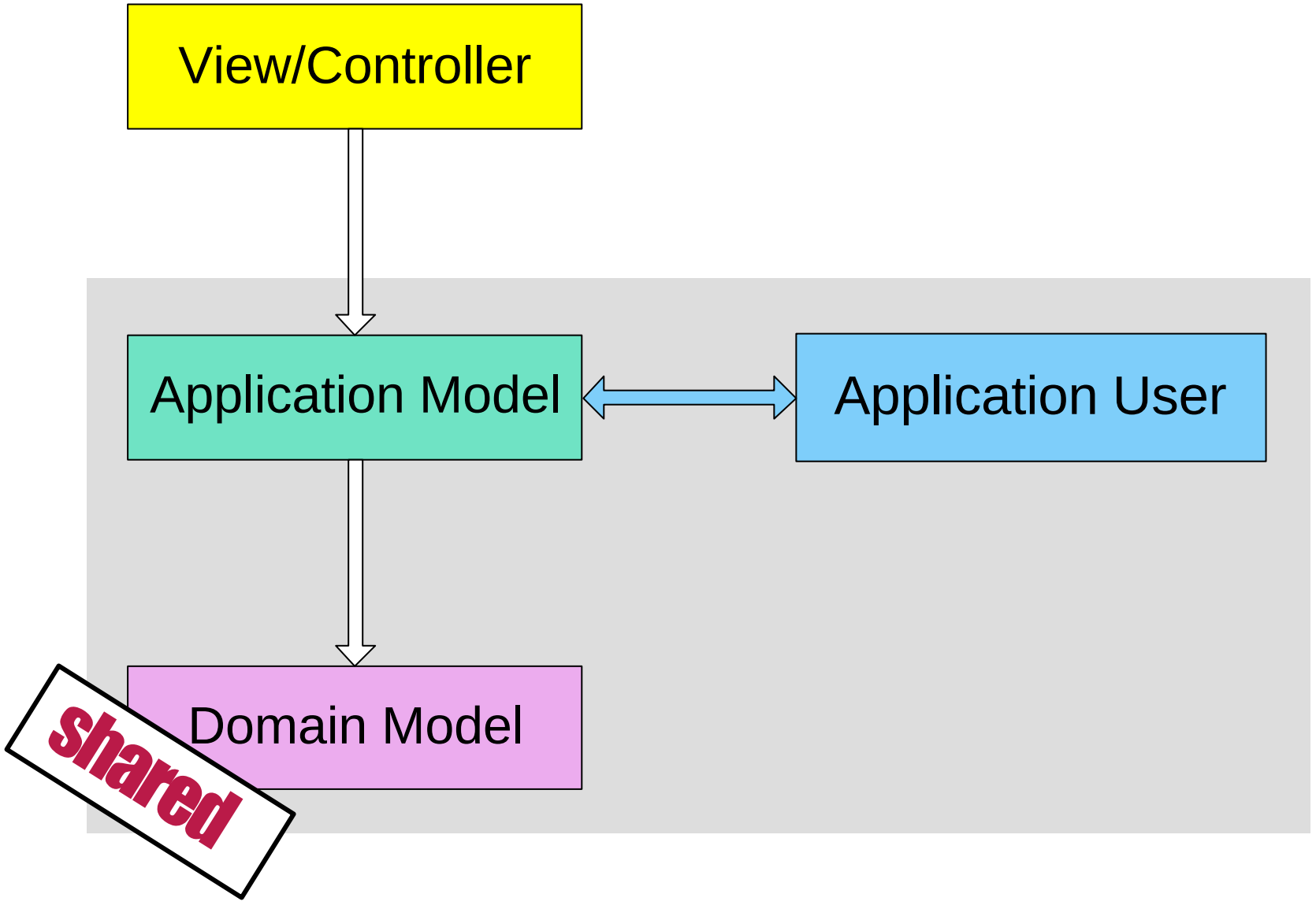
Support the development of

- object oriented,
- synchronous,
- interactive, and
- complex (e.g. hypermedia applications)

groupware.

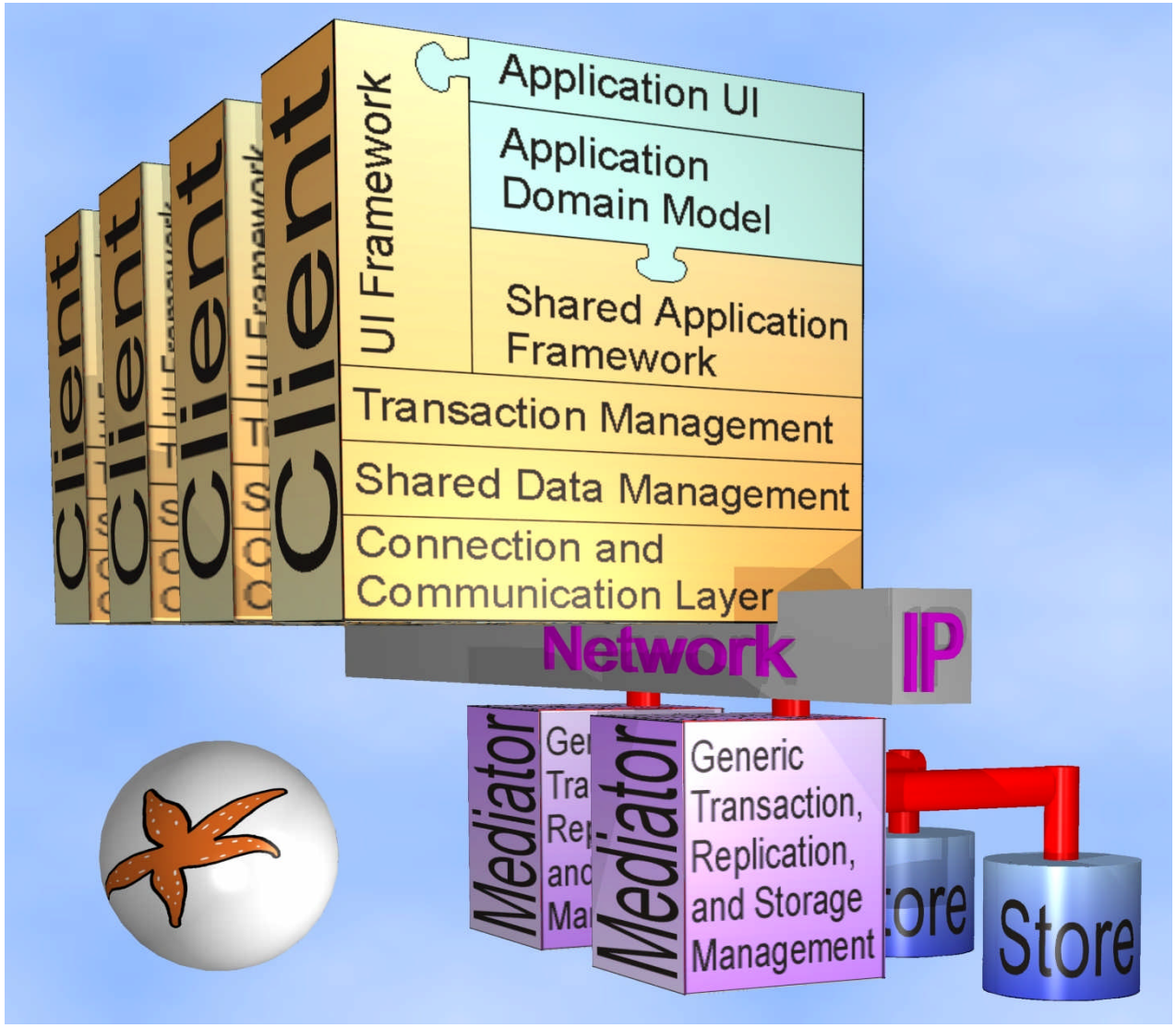
How COAST applications are structured

intelligent views



The COAST Framework

intelligent views



COAST in K-Infinity

intelligent views

point out functionality provided by COAST

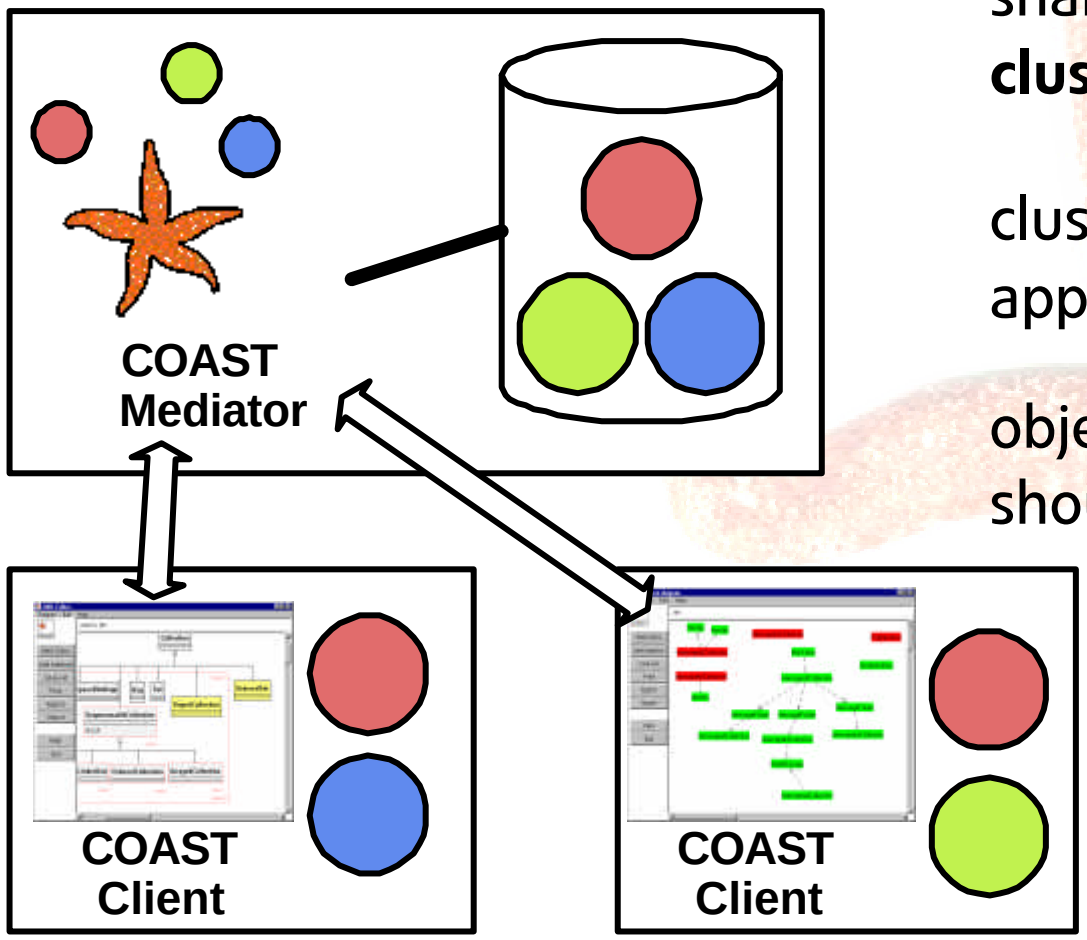
- replication & persistence
- concurrency control
- frame system
- view programming

explain how K-Infinity makes use of the functionality

demonstrate the functionality

Persistency and Replication

intelligent views



shared objects are bundled in **clusters**

clustering is critical for application performance

objects that are used together should be in the same cluster

Persistency in K-Infinity

intelligent views

using the COAST features

clustering strategy

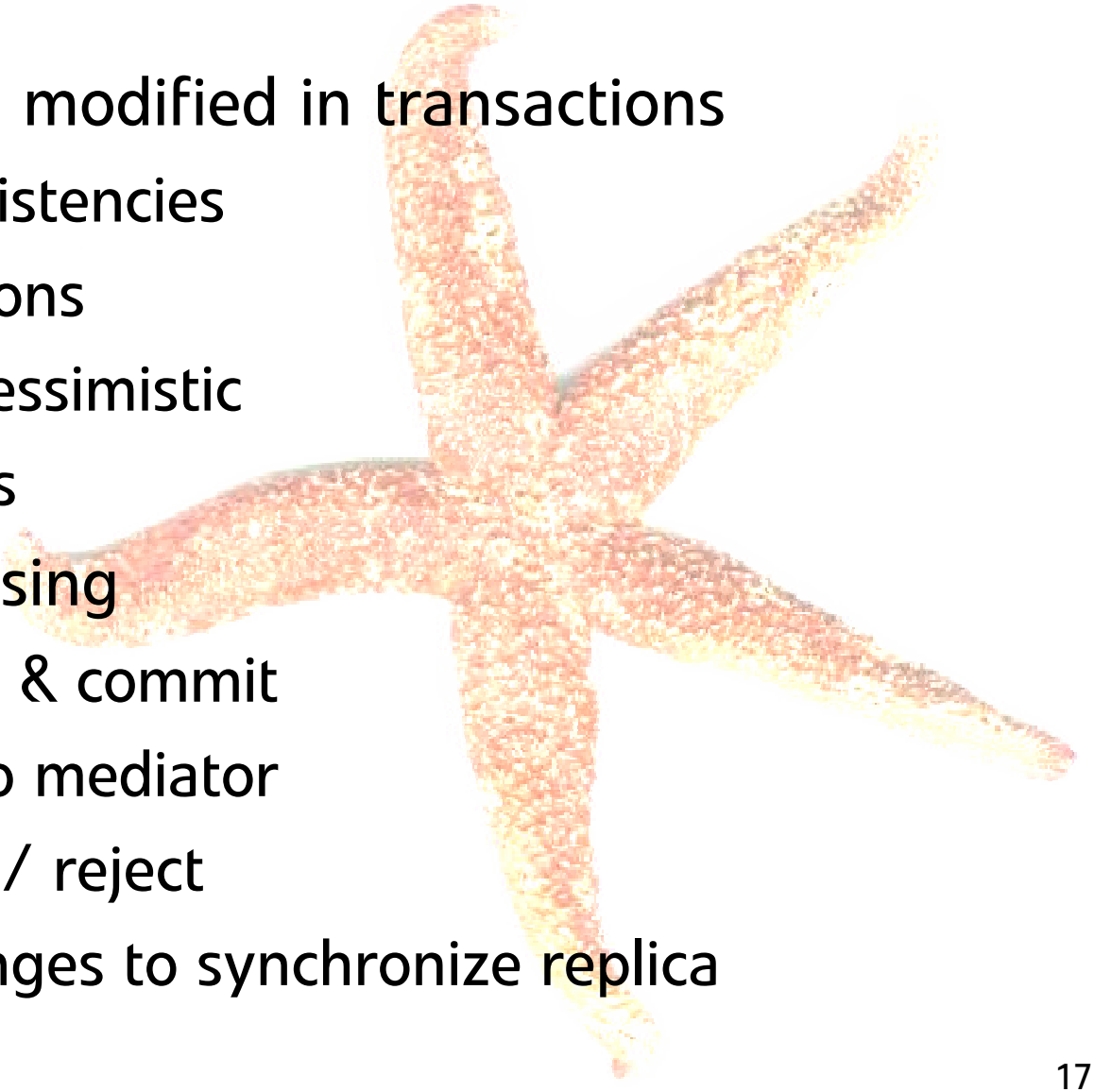
- COAST default behavoiur led to bad results
- open issue: re-clustering of COAST volumes

persistency is achieved

- implicitly by inhertance (COAST standard)
- explicitly by a method call (added for K-Infinity)

Transactions and Concurrency Control

intelligent views



shared objects are modified in transactions

- prevent inconsistencies
- short transactions
- optimistic or pessimistic
- ACID properties

transaction processing

- local execution & commit
- send agenda to mediator
- global commit / reject
- broadcast changes to synchronize replica

Concurrency Control in K-Infinity

intelligent views

experiences in K-Infinity

- use optimistic transactions wherever possible
- very little concurrency conflicts
  - separated responsibilites for each user
  - instant visibility of other user's actions

standard approach: locking

- single operations can influence the whole net
- generous locking in extreme leads to single user applications

DEMO: Concurrency Control

intelligent views

trigger a concurrency conflict

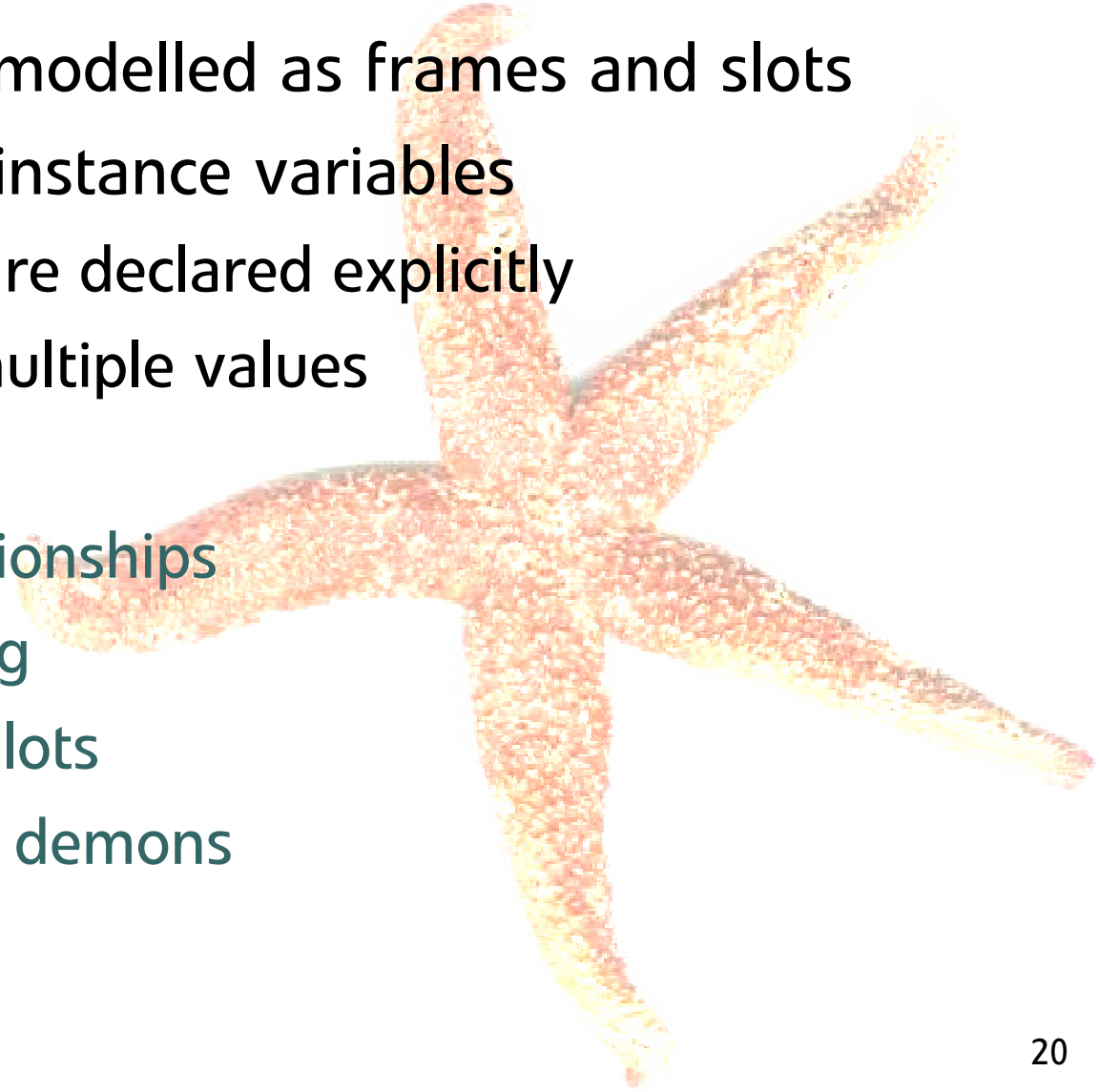
COAST Object Behaviour Framework

intelligent views

shared objects are modelled as frames and slots

slots are similar to instance variables

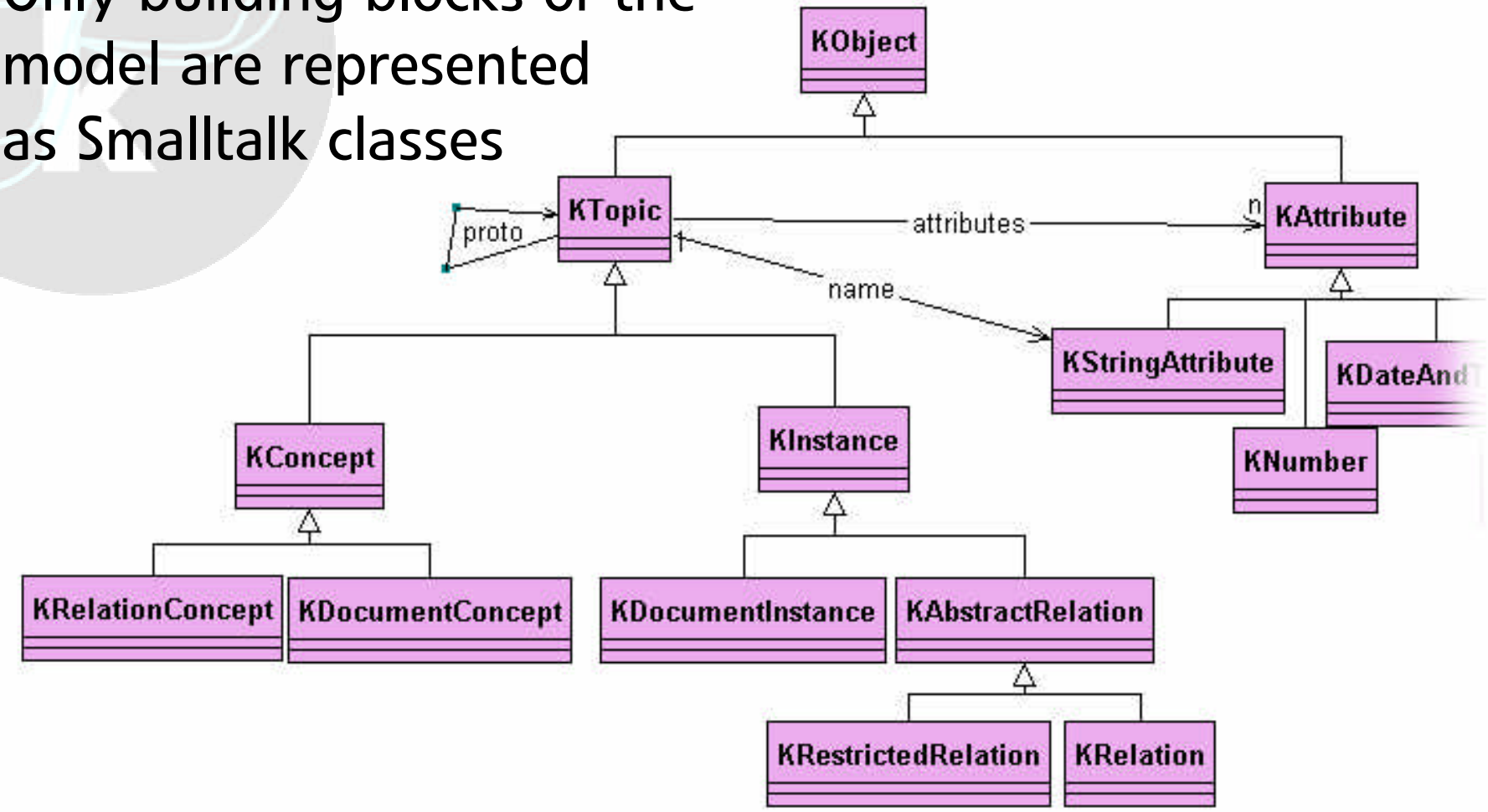
- slot properties are declared explicitly
- slots can hold multiple values
- demons
  - inverse relationships
  - type checking
  - constituent slots
  - user defined demons



The K-Infinity domain model

intelligent views

- Prototype system built on top of COAST
- Only building blocks of the model are represented as Smalltalk classes



K-Infinity Model: Pros and Cons

intelligent views

advantages

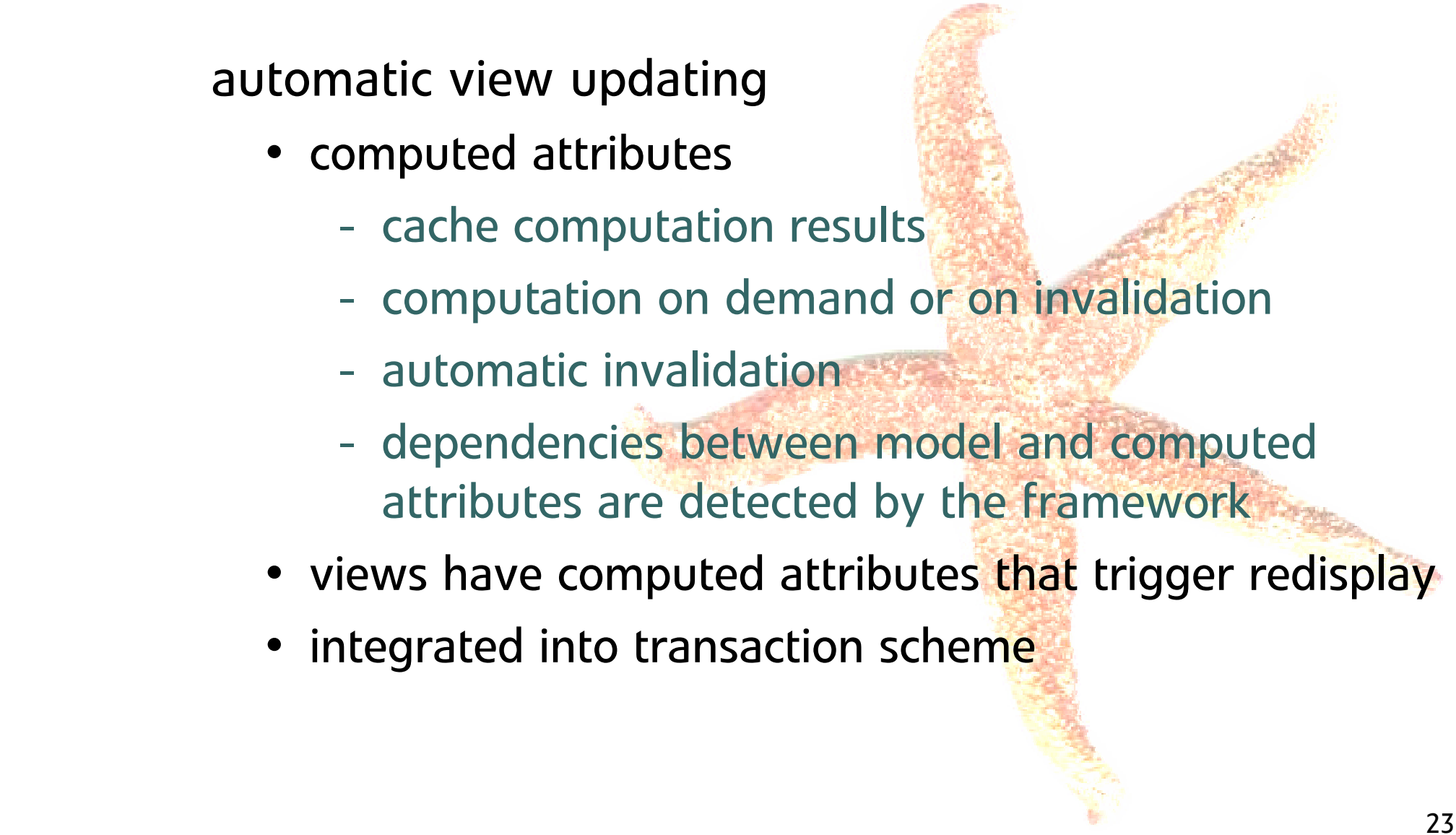
- very flexible
- no code changes for adaption to specific knowledge domain

disatvantages

- stuff had to be re-done that is already present in ST (e.g. inheritance – but it is multiple now ;-))
- not as fast as „native“ ST classes would be

UI Programming in COAST

intelligent views



automatic view updating

- computed attributes
  - cache computation results
  - computation on demand or on invalidation
  - automatic invalidation
  - dependencies between model and computed attributes are detected by the framework
- views have computed attributes that trigger redisplay
- integrated into transaction scheme

UI Programming in K-Infinity

intelligent views

COAST mechanisms work well

- display keeps consistent
- no explicit update notification code

complex visualization

- many dependencies
- wide parts are custom build due to requests of our design departement

hard work where COAST and standard mechanisms meet each other

DEMO: UI behavior in K-Infinity

intelligent views

change name of a topic & point out the change of all visualizations

change 'allows instances' flag and point out different visualization in the graph editor

And where is the Groupware?

intelligent views

K-Infinity shows only little of it

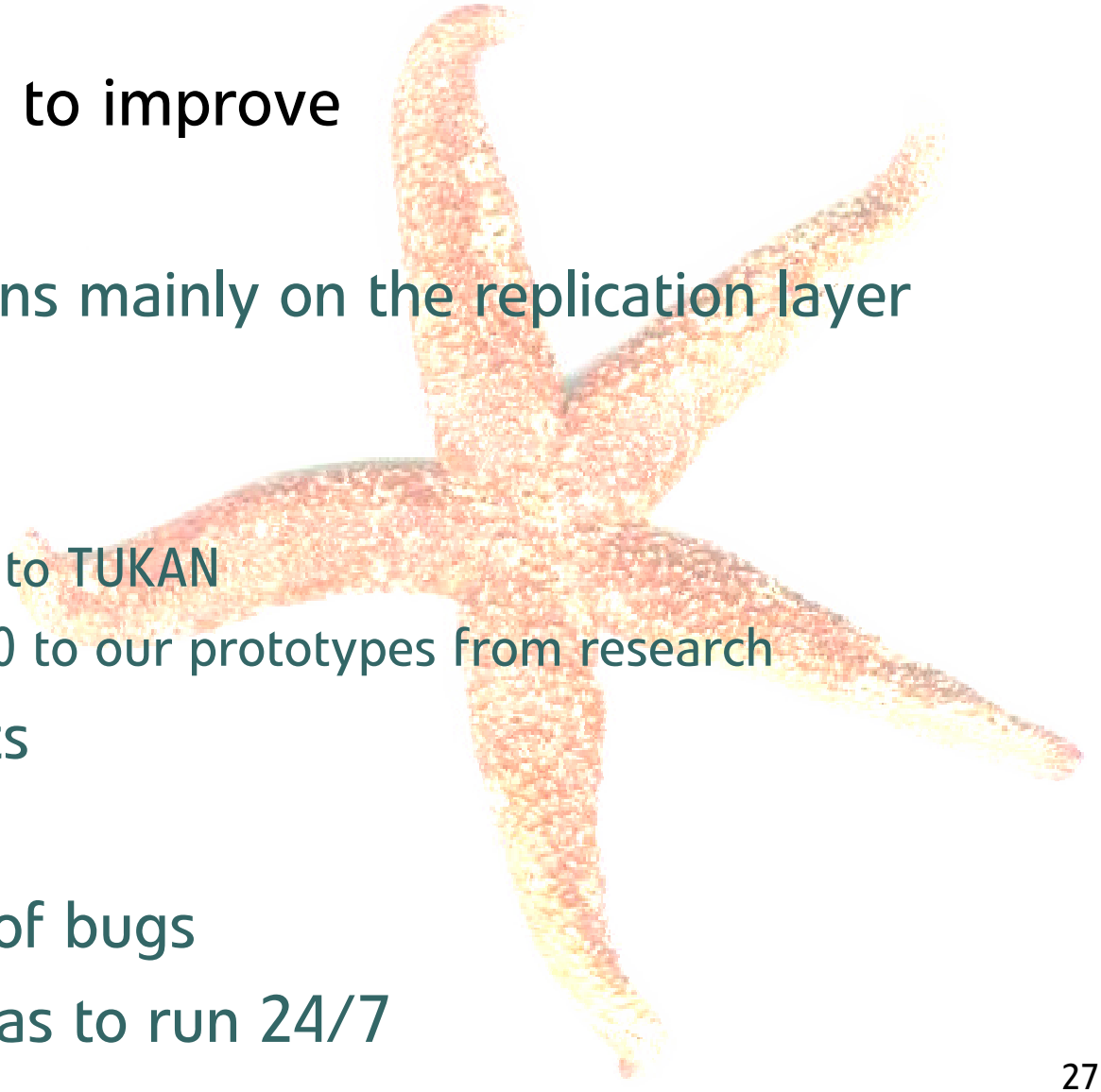
- groupware functionality hidden in the application
  - shared workspace
  - user list
  - chat
  - awareness

K-Infinity is used as a groupware application

- distributed users
- demand for co-ordination
- clients are asking for workflows

From Research to Industry

intelligent views



what did we need to improve

- performance
  - optimizations mainly on the replication layer
- scaleability
  - more data
    - factor 15 to TUKAN
    - factor 200 to our prototypes from research
  - more clients
- quality
  - found lots of bugs
  - mediator has to run 24/7

further information

intelligent views

www.i-views.de

www.opencoast.org