Interactions, transactions, and an emergent "web of models"

Sotiris Moschoyiannis

Talk **@ Informatics and Telematics Institute**

Centre for Research and Technology, Thessaloniki, Hellas

Monday 29.03.10

Outline

- Background (and disclaimer)
- Patterns of interaction
  - concurrent, distributed
- Long-running transactions
- Emerging network structures that support these
  - How local interactions come into play
  - Characteristics / features that draw upon ecosystem concepts
- Exposing models -- rules-based approach
- Future directions

Patterns of interaction..

Modelling behaviour

Adaptation of *vector languages* [Shields 1979, 1997] which use tuples of sequences, one for each sequential subsystem.

- Multiple access points
  - distribution, concurrency

- Concurrency via independence

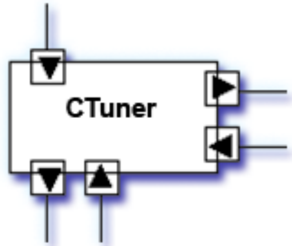
  ATS [Shields,1985], [Mazurkiewicz,1988]

- Composition as in process algebras

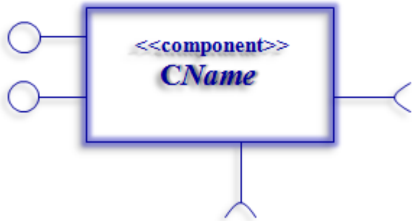
  CSP [Hoare,1985], CCS [Milner,1980]

- Operations performed coordinate-wise

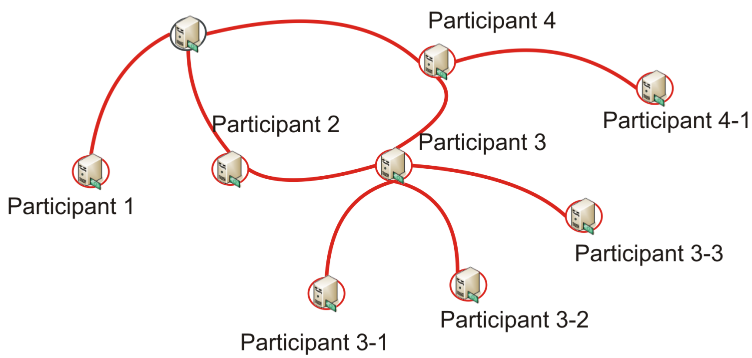
  concatenation, prefix-ordering, right-cancellation



Koala

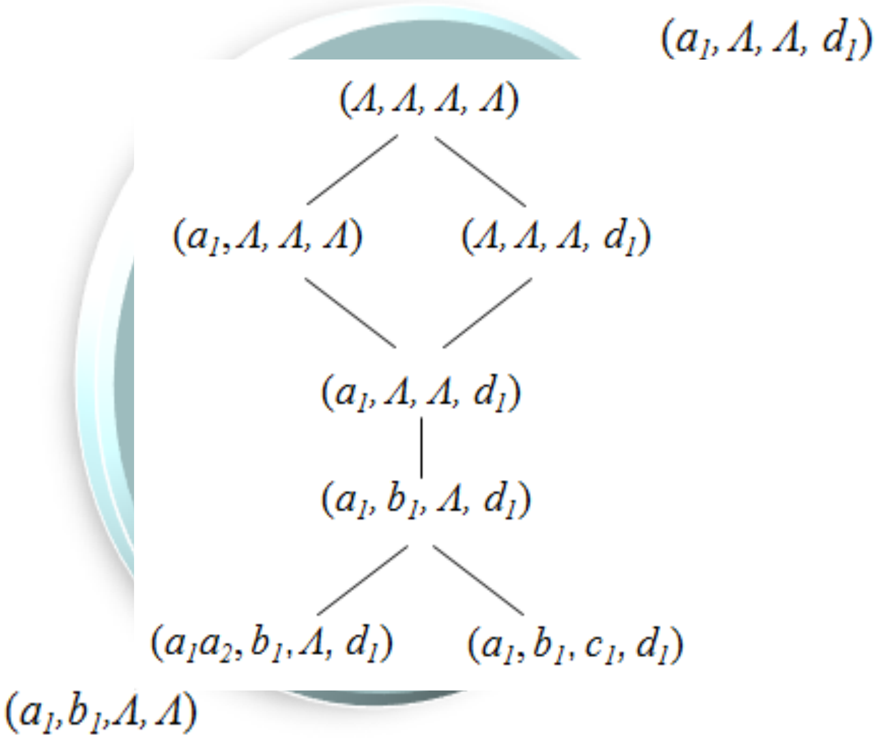


UML



Distributed P2P network

Patterns of interaction



- Record observable behaviour
- Restrict to allowed sequences of events
- Exploit the algebraic properties -- formal analysis prior to deployment
- Keep 'history' of dependencies in the interaction

Order structure – *building blocks*

$(s1, s2, \Lambda)$

$(s1, \Lambda, \Lambda)$

$(\Lambda, \Lambda, \Lambda)$

$s1$ before $s2$

appear in
ordered
vectors

$(s1, \Lambda, \Lambda)$ $(\Lambda, s2, \Lambda)$

$(\Lambda, \Lambda, \Lambda)$

$s1, s2$ are alternative

incomparable vectors
from the same vector,
not leading to a common
vector

$(s1, s2, \Lambda)$

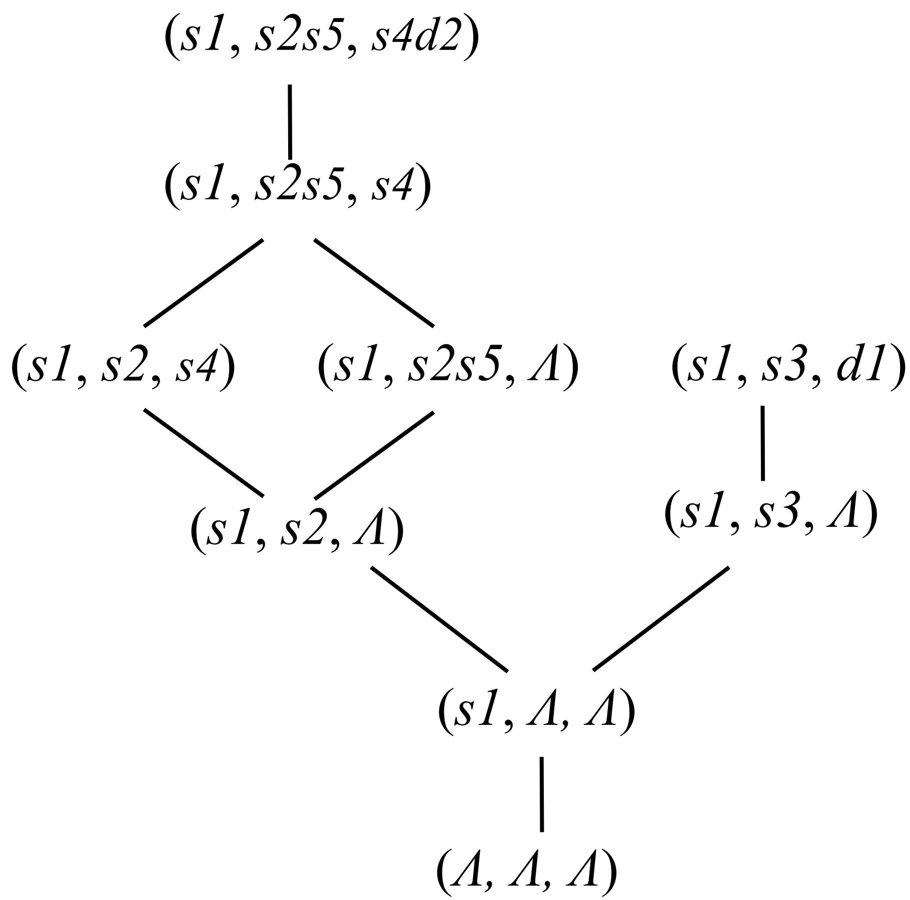
$(s1, \Lambda, \Lambda)$ $(\Lambda, s2, \Lambda)$

$(\Lambda, \Lambda, \Lambda)$

$s1, s2$ are concurrent

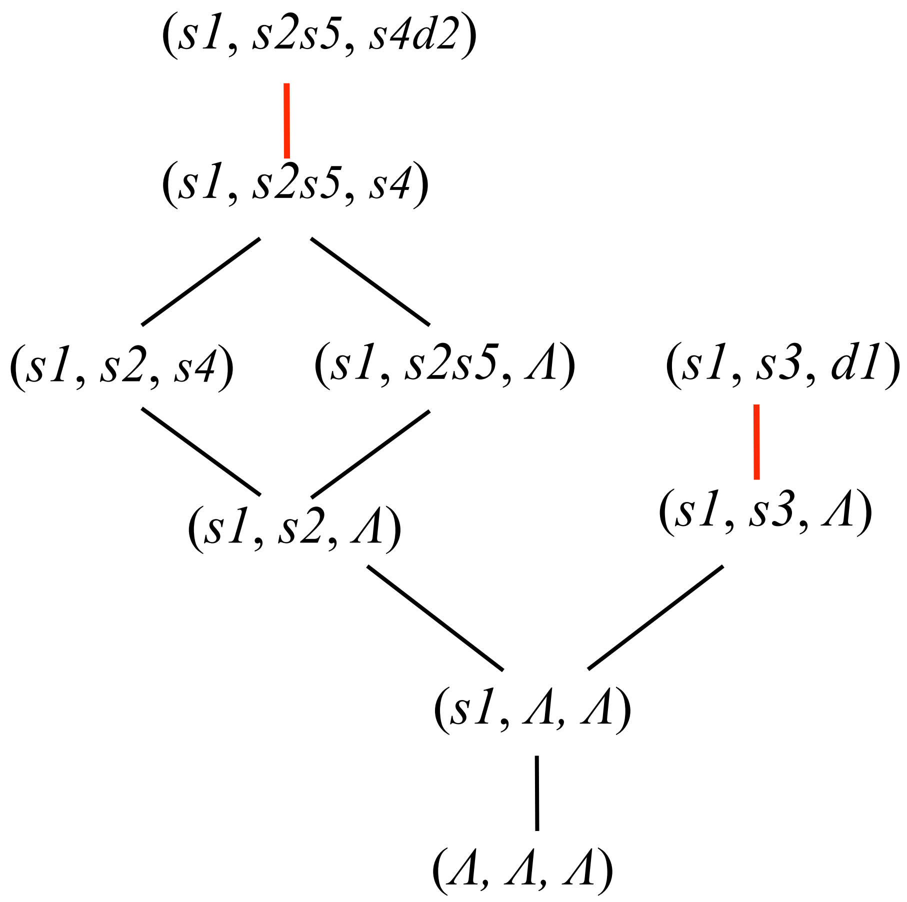
incomparable
independent vectors
from the same vector,
leading to a common vec

Order structure



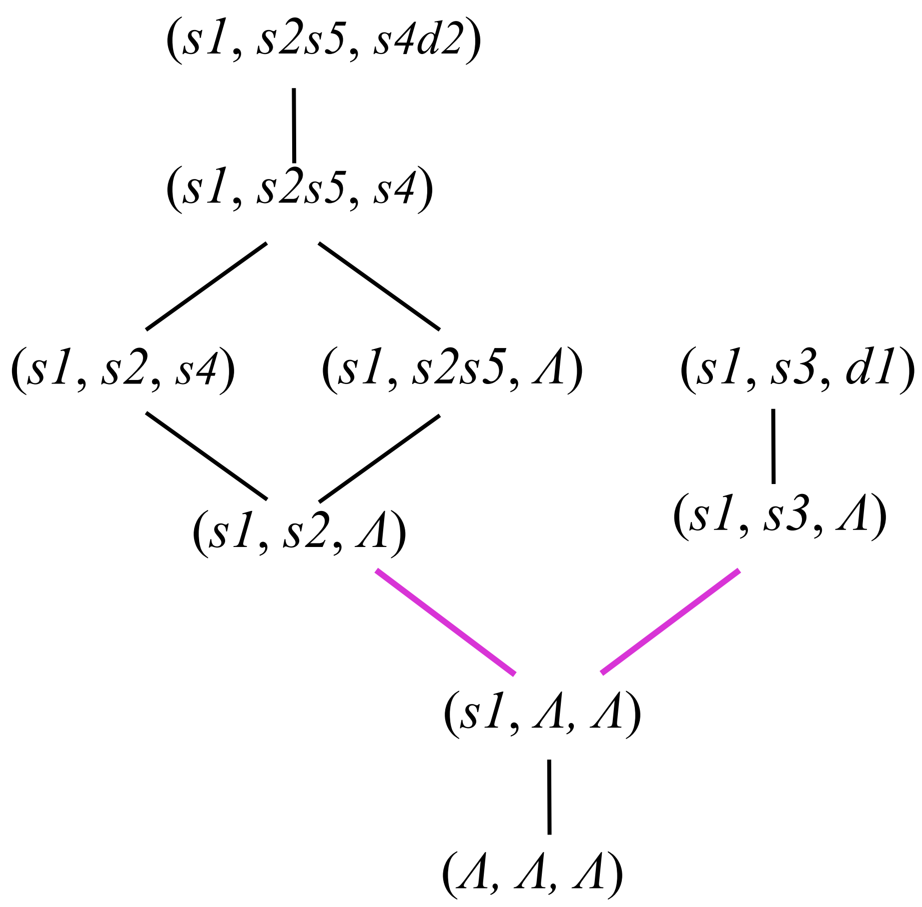
Using the building blocks… like legos!

Order structure - *sequential*



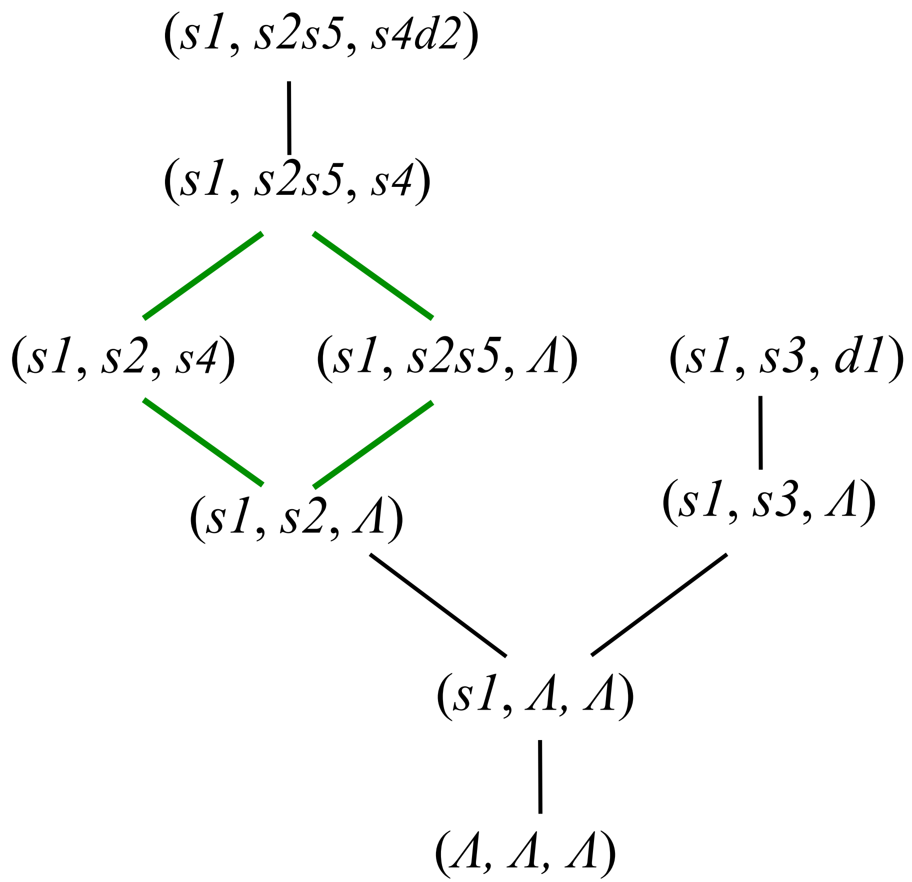
d1 occurs only after *s1* and *s3* have occurred

Order structure - *alternative*



after *s1* there is a choice between *s2* and *s3*

Order structure - *parallel*



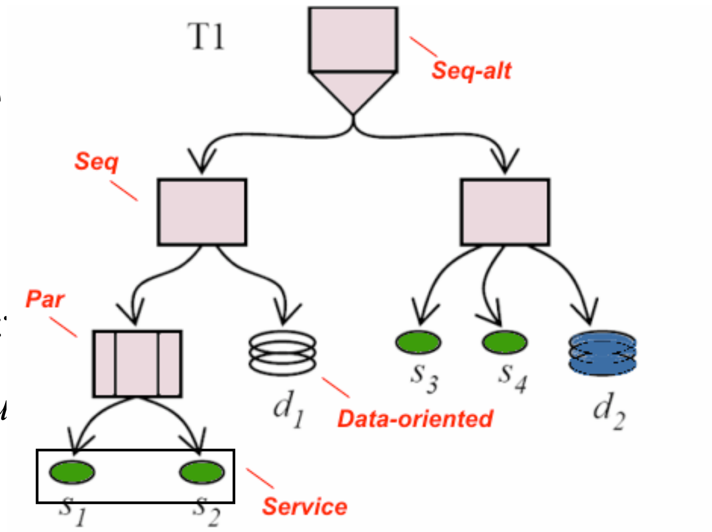
after *s1* and *s2* have occurred, *s4* and *s5* happen concurrently

Concurrency

- Independence relation on actions – binary relation
  - $-\ a\,\iota\,b \Rightarrow b\,\iota\,a$
  - $-\ a\,\iota\,b \Rightarrow a \neq b$

- Generates equivalence relation on sequences of ac

  $$x \equiv_{\iota} y \Rightarrow \exists u,v \in A^*, \exists a,b \in A : a\,\iota\,b \wedge x = uabv \wedge y = u$$

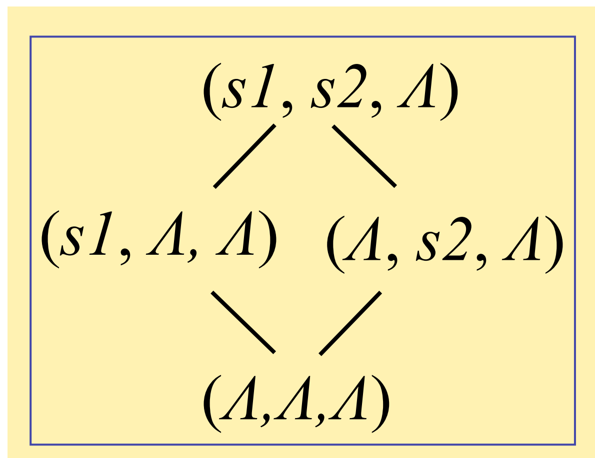


- Independence relation on (lifted to) vectors

  $$\underline{u}\ ind\ \underline{v} \Leftrightarrow \forall t \in T, \underline{u}(t) > \Lambda \Rightarrow \underline{v}(t) = \Lambda$$

- Models true concurrency -- $\underline{a}_1$ and $\underline{a}_2$ are concurrent iff

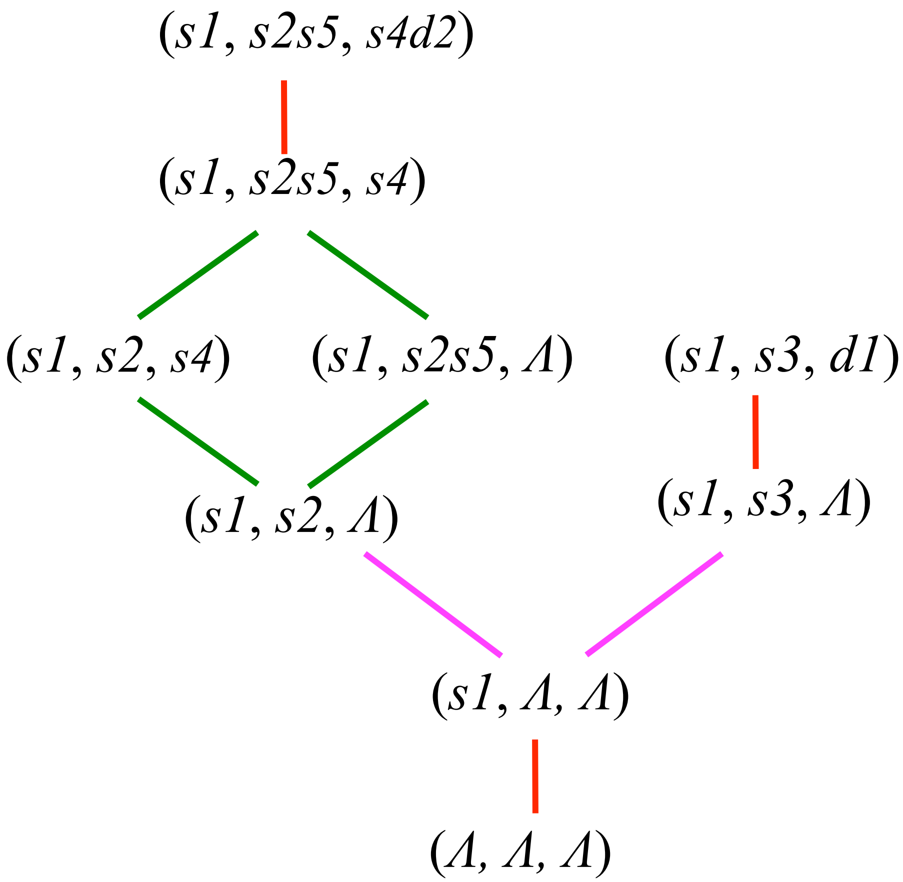
  $$\underline{a}_1\ ind\ \underline{a}_2 \quad \text{and} \quad \underline{u}.\underline{a}_1.\underline{a}_2 = \underline{w} = \underline{u}.\underline{a}_2.\underline{a}_1$$



Non-interleaving models of concurrency are due to Shields [Shi85, Shi97] *and Mazurkiewicz* [Maz88]

Transaction (vec) language

- Dependencies manifest themselves in the resulting order structure



Discreteness

- **Finiteness** -- only a finite number of actions/events may occur within finite time
- Excludes ascending or descending chains of occurrences of events (Zeno's paradoxes)
- In **discrete** systems, events do not blur into one another

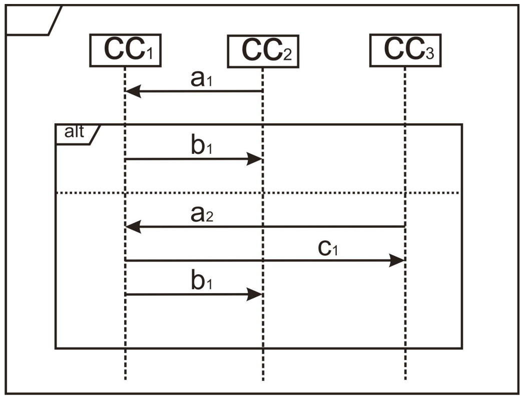
A transaction language V is *discrete* iff $\underline{\Lambda} \in V$ and whenever $\underline{u}, \underline{v}, \underline{w} \in V$ such that $\underline{u}, \underline{v} \leq \underline{w}$ then, $\underline{u} \prod \underline{v} \in V$ and $\underline{u} \coprod \underline{v} \in V$.

Local left-closure

- Every **earlier part** of a behaviour is itself a behaviour
- Local as it is applied on each coordinate…

A transaction language V is locally left-closed iff whenever $\underline{u} \in V$ and $t \in T$ and $x \in \beta(t)^*$ such that $\Lambda < x < \underline{u}(t)$ then, $\exists\, \underline{v} \in V$ such that $\underline{v} \leq \underline{u}$ and $\underline{v}(t) = x$.

Formal reasoning on interaction scenarios



Suppose that $c = (\Sigma, V)$ is represented in a seq diag by a lifeline $Cline = (c, Loc, l_0, Op_\Sigma, SE, RE, Path)$. We define an injective function $vec\_map : Loc' \to \wp(V_\Sigma)$ given by

- $vec\_map(l_0) = \underline{\Lambda}_\Sigma$
- $vec\_map(l) = \{\underline{v}_l^{(1)}, \underline{v}_l^{(2)}, \ldots, \underline{v}_l^{(m)}\}$ where $m = |vec\_map(\tilde{l})|$ and $\tilde{l} \in Loc$

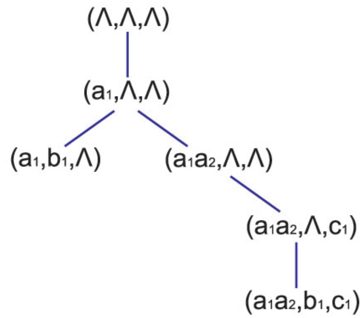
  such that $time(\tilde{l}) = time(l) - 1$ and for each $j, 1 \le j \le m$,

$\underline{v}_l^{(j)} = (v_{l_1}^{(j)}, v_{l_2}^{(j)}, \ldots, v_{l_n}^{(j)})$

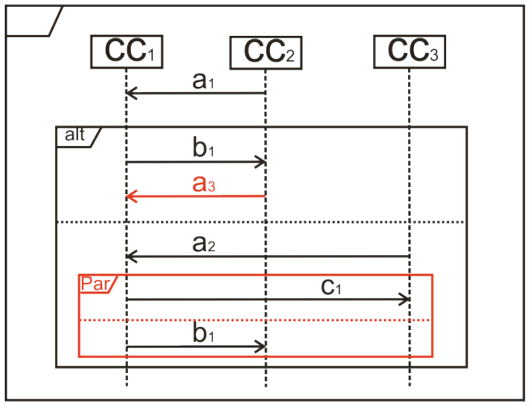
where n is the number of ports of c and each coordinate is given by

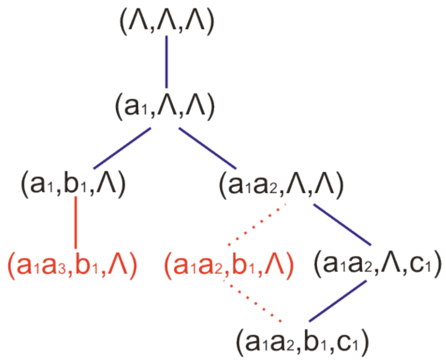
$$v_{l_p}^{(j)} = \begin{cases} v_{\tilde{l}_p}^{(j)}.e & , ((l,e) \in SE \vee (e,l) \in RE) \wedge e \in \beta(p) \\ v_{\tilde{l}_p}^{(j)} & , \text{otherwise} \end{cases}$$

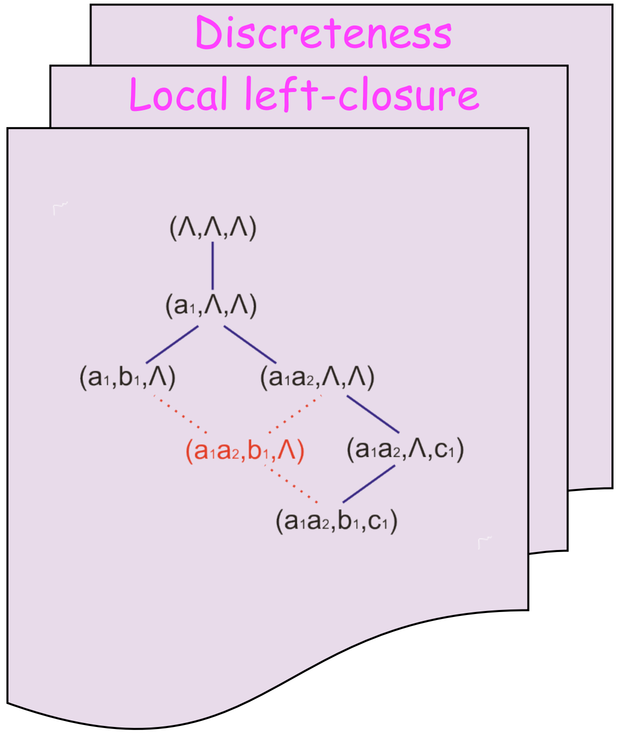
where $1 \le p \le n$.



UML interaction diagram



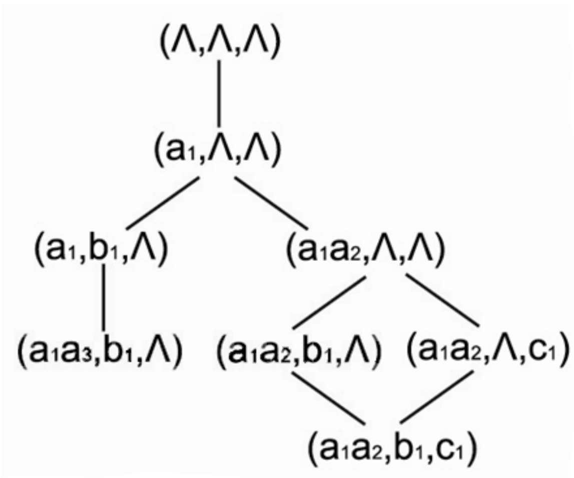


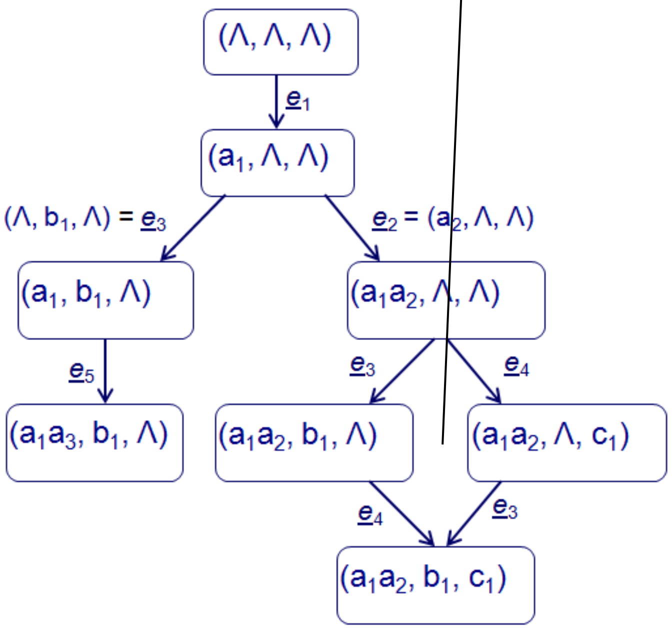


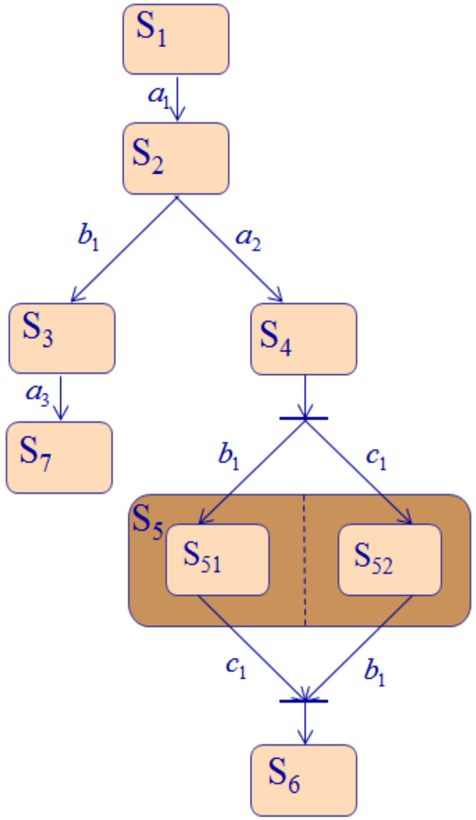
Moschoyiannis, Razavi, Krause. Transaction Scripts: Making implicit scenarios explicit. *ETAPS-FESCA'08, ENTCS, Elsevier*, 2008. *To appear*

Algebraic auto

applied in model checking
[Kwiatkowska,Norman,Parker,2006]







- Order-theoretic structure preserved
- Express true-concurrency in UML
- Interesting algebraic properties
  e.g. symmetries -> cellular pathways

UML state diagram

Moschoyiannis, Shields, Krause. Modelling Behaviour with Concurrent Automata. In *ETAPS 2005 -FESCA'05 ENTCS, 141(3): 199-220, Elsevier, 2005.*

Temporal properties of the interaction

- Distributed temporal logic (MDTL) interpreted over concurrent automata

$$C_{L_v} ::= \{c.H_c\}_{c \in \mathrm{C}_v} \mid C_v$$

$$H_c ::= ATOM_c \mid \neg H_c \mid H_c \Rightarrow H_c \mid H_c \bigcup H_c \mid H_c \,\Delta\, H_c$$

$$C_v ::= c.Mes!d \leftrightarrow d.Mes?c \mid c.Mes!d \rightarrow d.Mes?c \mid H_{obs}$$

$$ATOM_c ::= true \mid Att\ \theta\ t \mid \triangleright\ Mes!d \mid \triangleright\ Mes?d \mid Mes!d \mid Mes?d$$

- Home logic -- individual participant's viewpoint

$P1.(m_2?P2 \;\; \Delta \;\; m_3?P3)$

- Communication logic -- interactions between participants

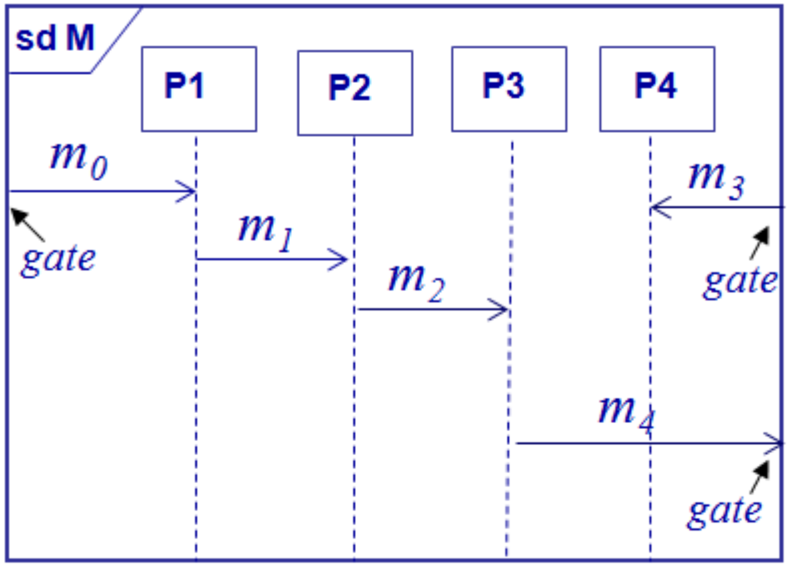
$P1.m_1!P2 \longrightarrow P2.m_1?P1$

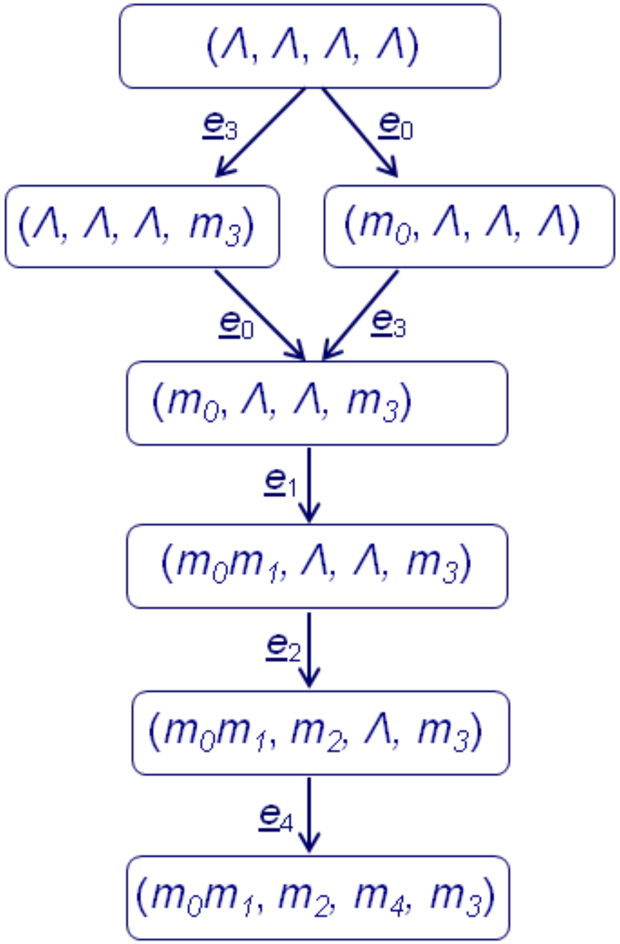
$P1.m_1!P2 \longleftrightarrow P2.m_1?P1$

Temporal properties of the interaction

- After m_1, some time in the future m_4 will happen

  P1.m_0?P2 ►P3.m_4!

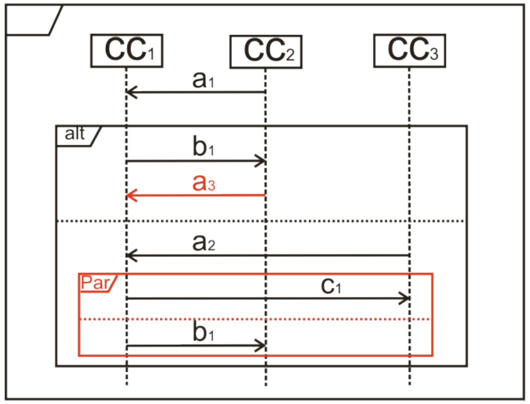




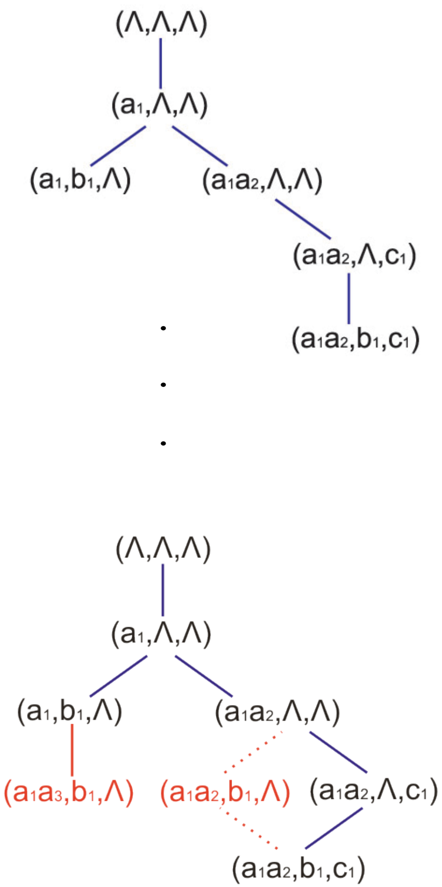
- Talk about liveness, fairness

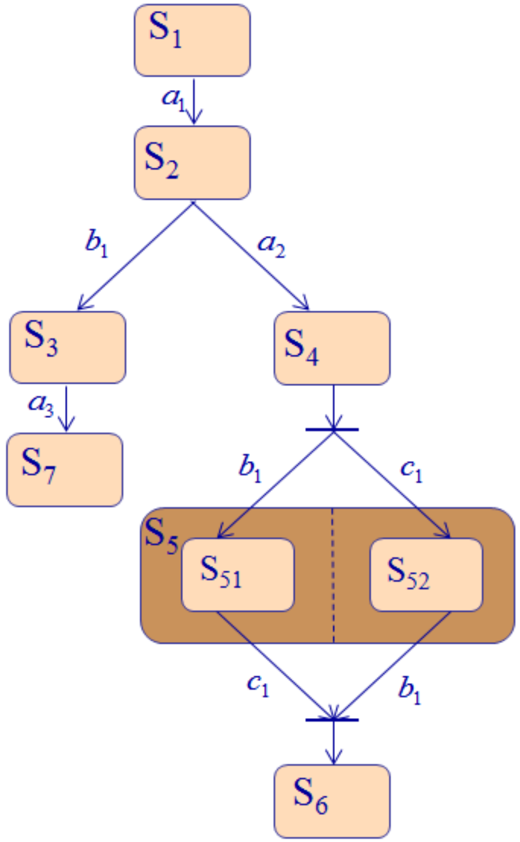
Bowles and Moschoyiannis. Concurrent Logic and Automata Combined. In *CONCUR 2006 – FOCLASA ENTCS, 175(2): 135-151, Elsevier, 2007.*

Formal translation of design models



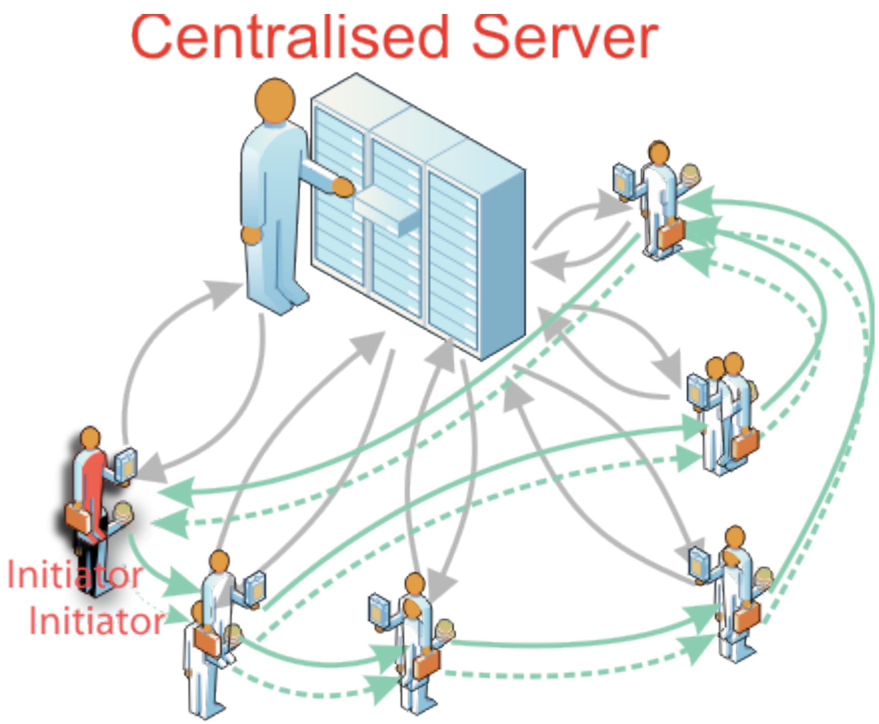
UML sequence diagram





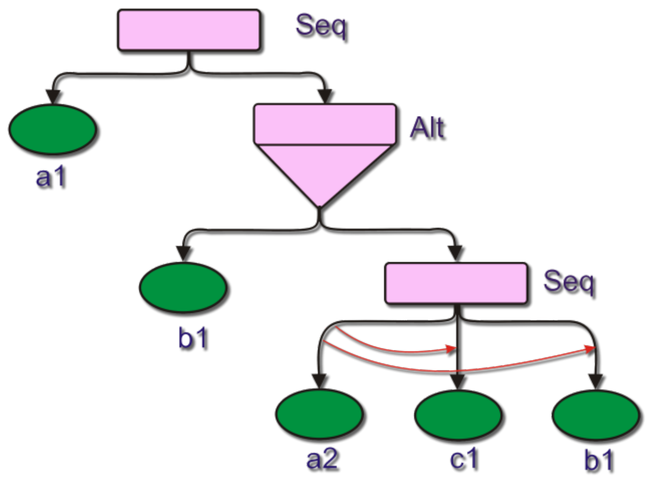
UML state diagram

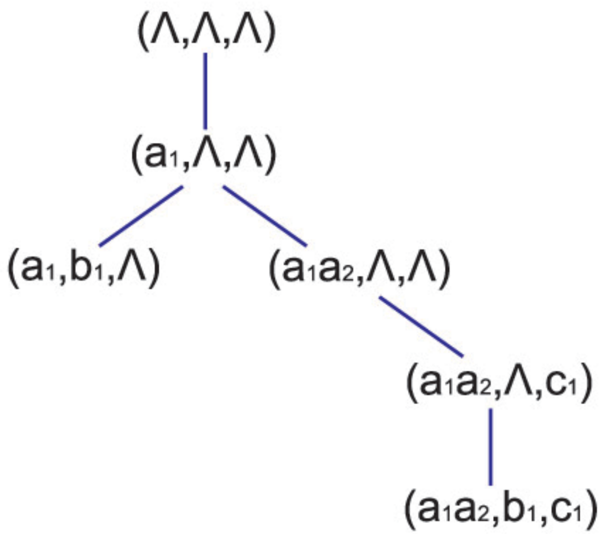
Transactions



- Correspond to long-term business activities
- Involve the execution of services
- Are complex interactions
- Dependencies within and across transactions
- Coordination of underlying services
- Compensation – execute *all or nothing*

Coordinating distributed transactions





Razavi, Moschoyiannis, Krause. A coordination model for distributed transactions in DEs. *IEEE-DEST 2007.*

- Determines participants and the required services
- Transaction context (tree) issued by Initiator
- Identify patterns service compositions should follow (forward behaviour)
- Compensate for previously successful (inter)actions, if some failure occurs (compensating behaviour)
- All participants' actions considered at each point during the interaction

Forward and compensating behaviour

- Derive sequences of compensating actions
- Hand
- Identify alternative paths of execution (for
- Pres (omitted results)
- No additional semantics!

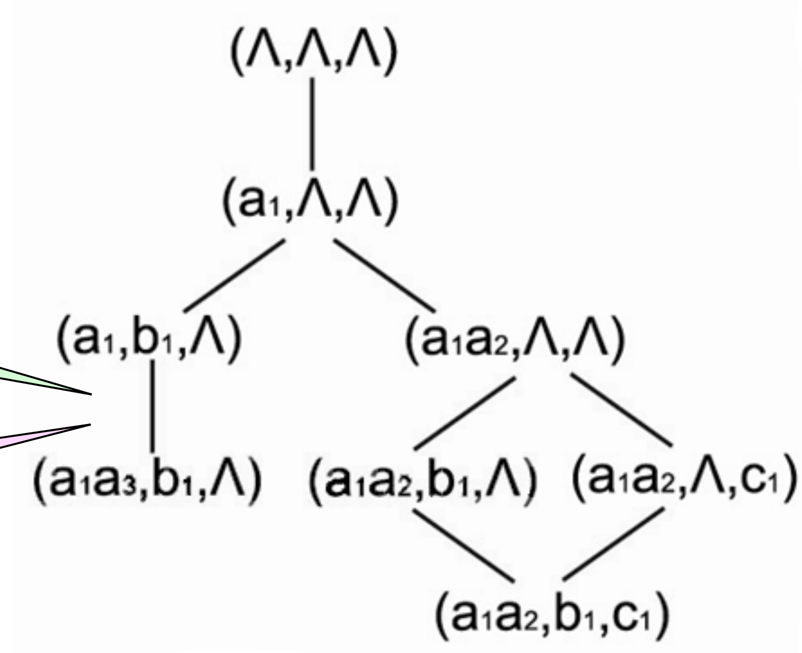
$(a_1, b_1, \Lambda).(a_3, \Lambda, \Lambda) = (a_1a_3, b_1, \Lambda)$

concatenation

$(a_1a_3, b_1, \Lambda) / (a_1, b_1, \Lambda) = (a_3, \Lambda, \Lambda)$;

$(a_1a_3, b_1, \Lambda) / (a_3, \Lambda, \Lambda) = (a_1, b_1, \Lambda)$

right-cancellation



Moschoyiannis, Razavi, Zheng, Krause. Long-running transactions: semantics, schemas, implementation. In *IEEE-DEST 2008, IEEE Computer Society, 2008.*

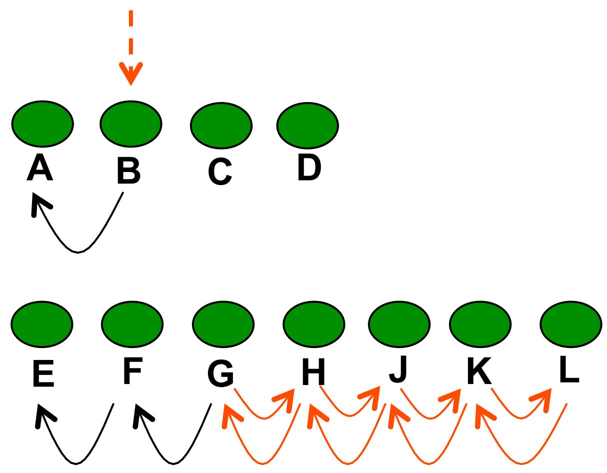
Concurrency and compensation

- Compensating CSP (c-CSP) [Butler, Hoare, Ferreira, 2006]
  - transactions modelled by sequential processes
  - no communication allowed, only synchronisation on terminal events
  - in concurrent execution, may lead to costly chains of rollbacks (in compensating)

- Compensation in flow composition languages [Bruni,Melgatti,Montanari, 2005]
  - also uses sequential processes
  - based on *Sagas* transactions [Garcia-Molina, Salem, 1987] –- linear, no nesting

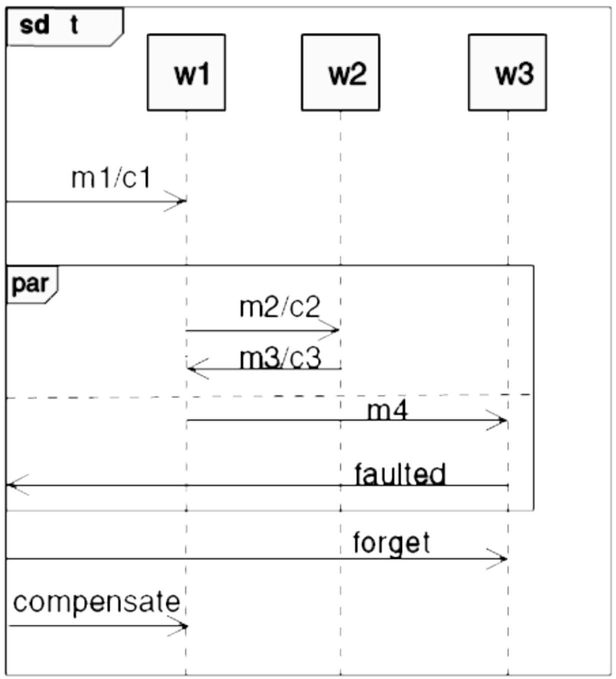
pp = <A, B, C, D, D^o, C^o, B^o, A^o,>

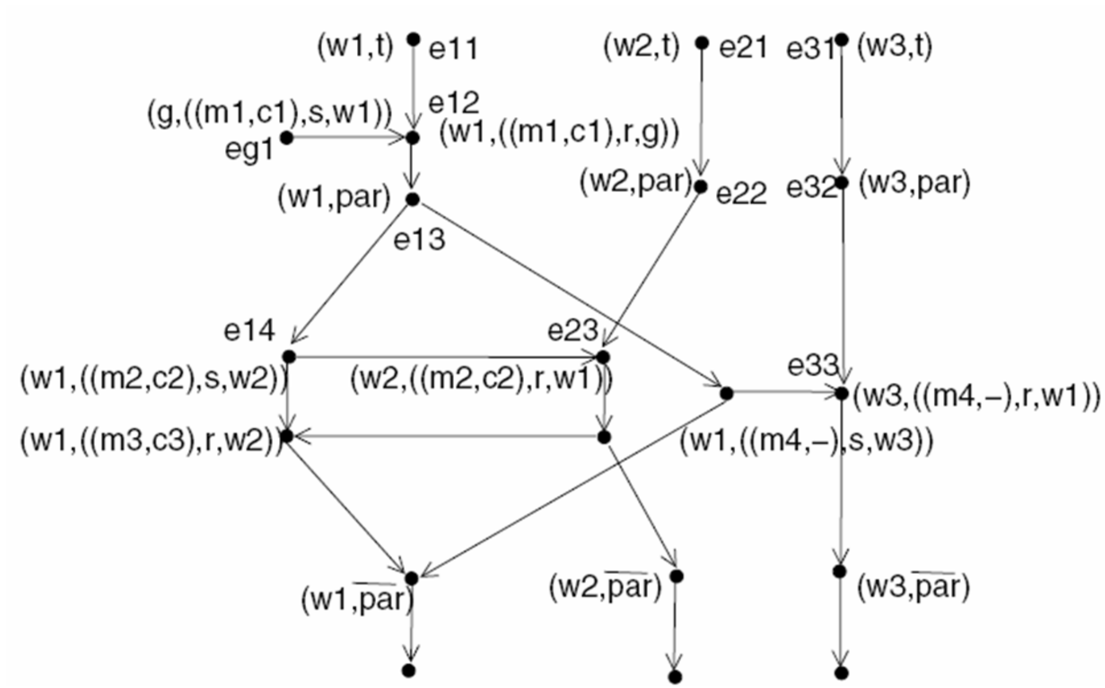
qq = <E, F, G, H, J, K, L, L^o, K^o, J^o, H^o, G^o, F^o, E^o>



Labelled event structures

- LES define relations on the set of events involved [Winskel, 1986] -- causality, non-determinism and, through these, concurrency
- To model forward and compensating behaviour we look into configurations, paths, transitions…

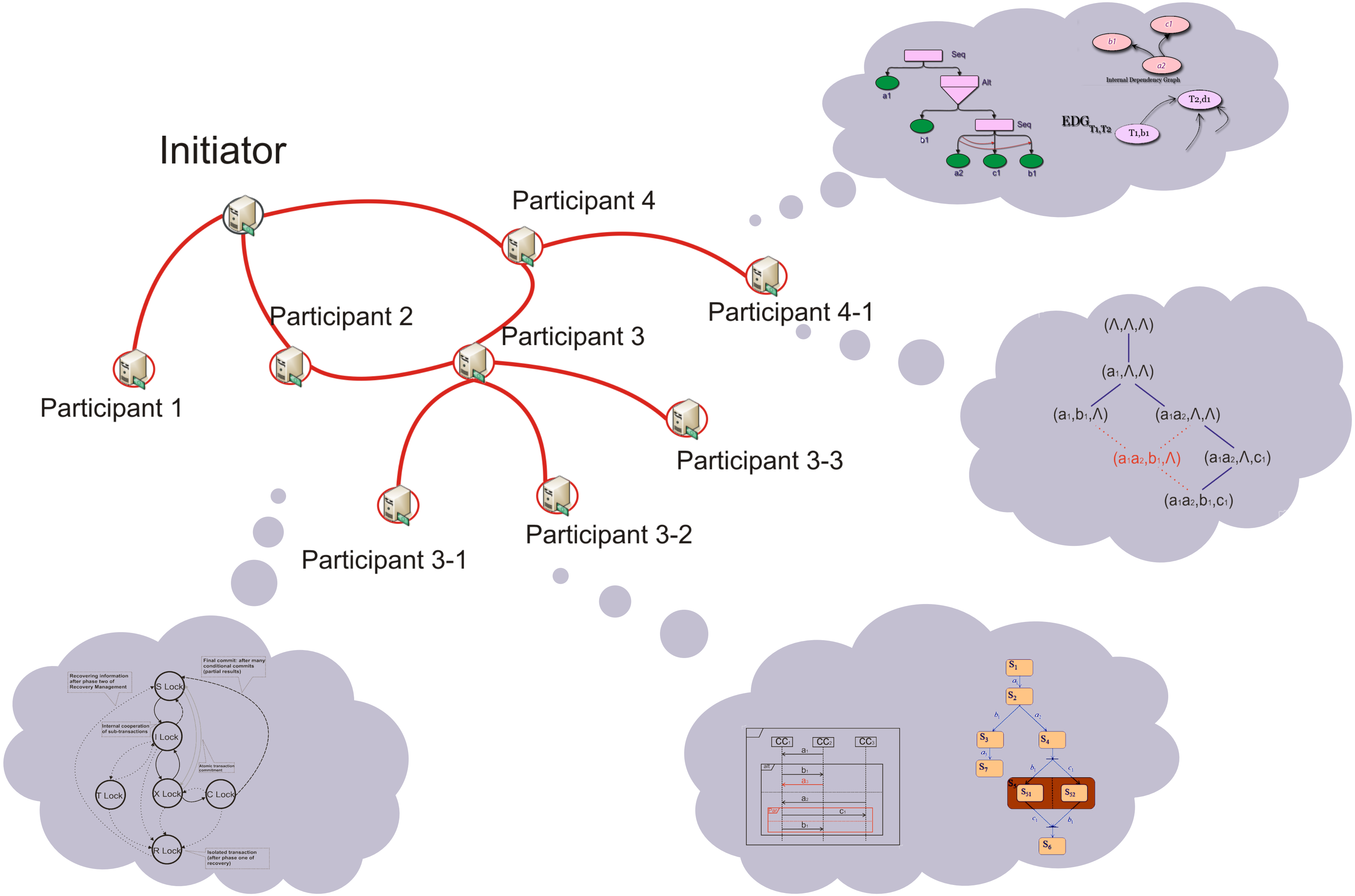




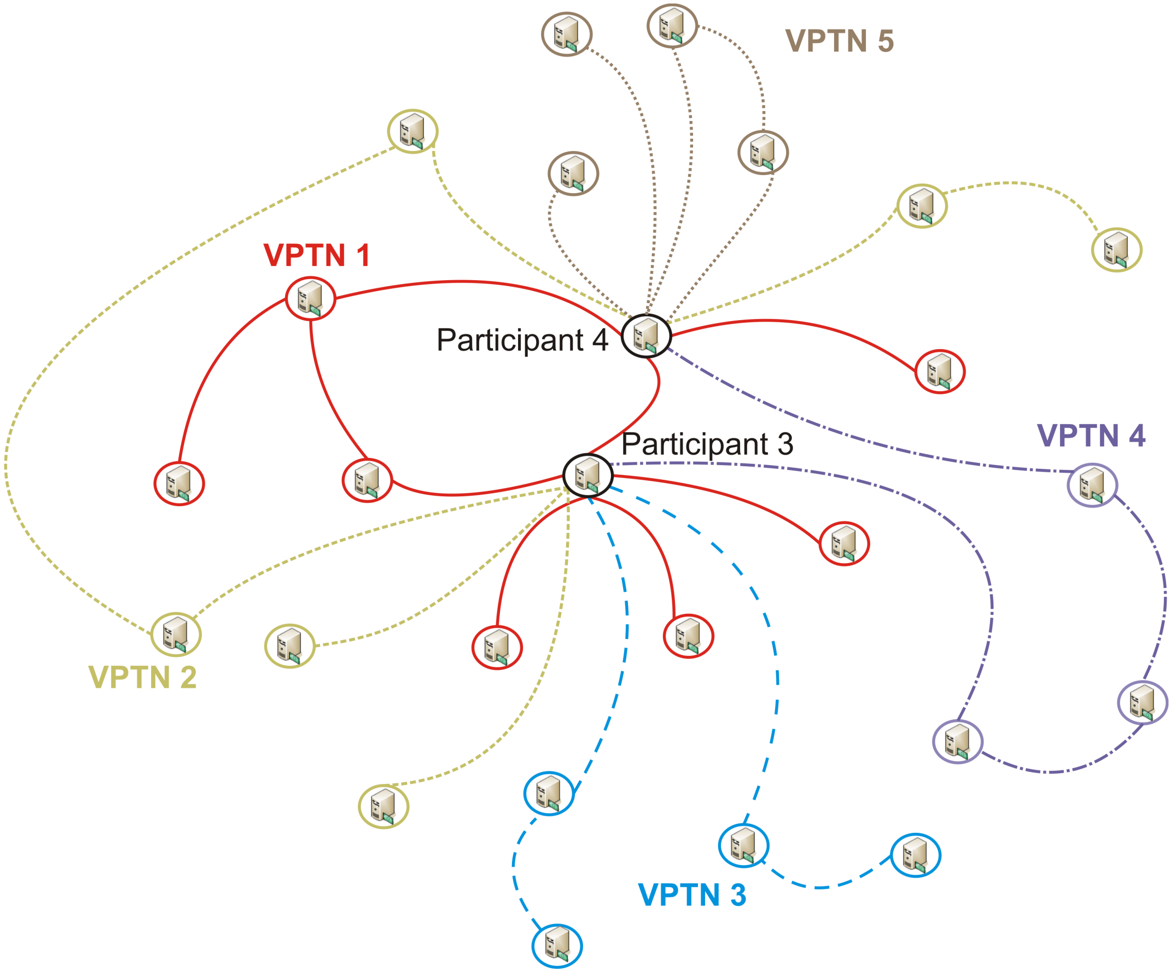
Bowles and Moschoyiannis. When things go wrong: interrupting conversations. In *ETAPS 2008 – FASE'08 LNCS 4961 ,pp. 131-145 , Springer, 2008.*

From local interactions to emerging network structures..

Local interactions



Emerging network



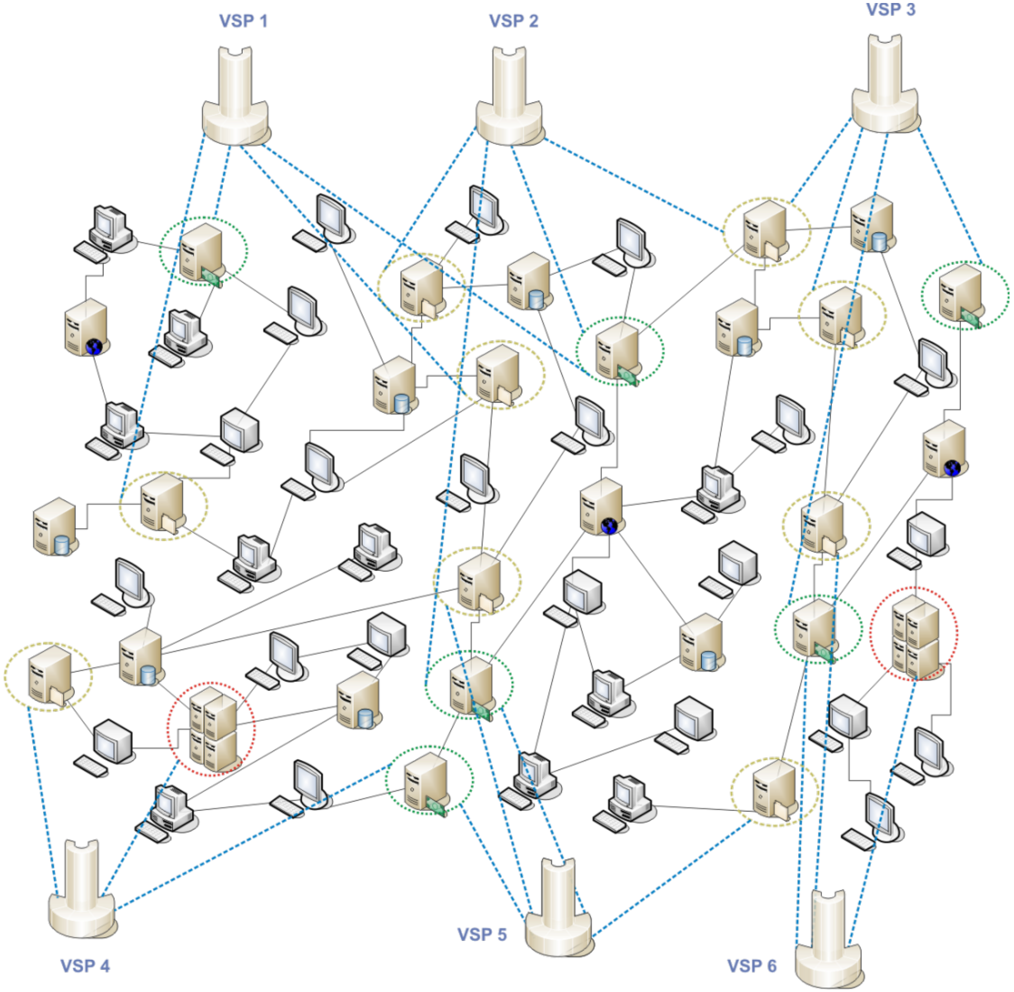
Razavi, Moschoyiannis, Krause. A scale-free business network for digital ecosystems. In *IEEE-DEST 2008, IEEE Computer Society, 2008.*

Best candidate for interconnecting

- Stability
  - Availability (during promised online time)
- Trust and accountability
  - Business activities
  - Community building
- Security..

Measured and assigned by neighbouring peers. Continuously.

Dynamic Virtual Super Peers

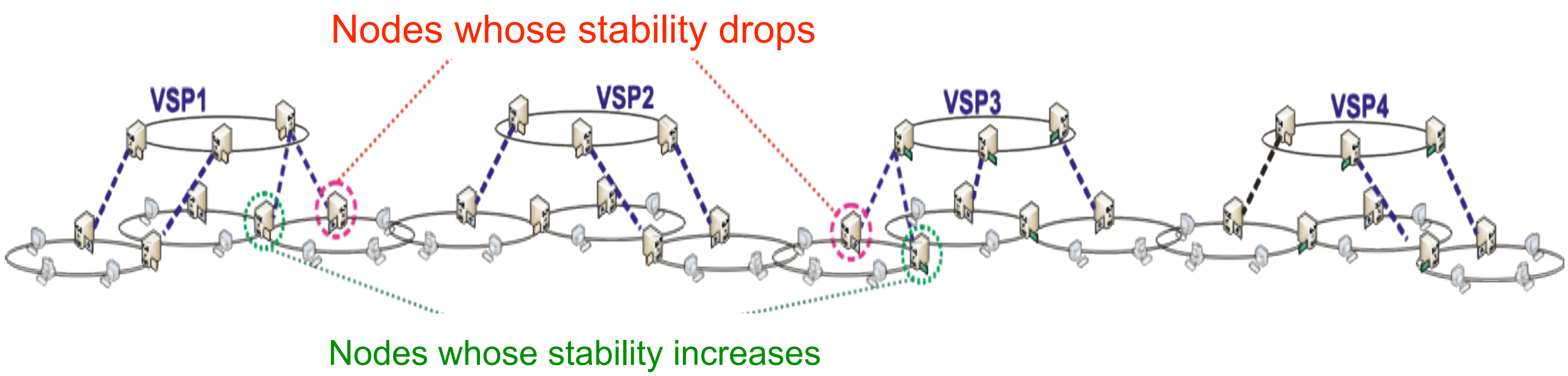


- Aggregations of most stable nodes
  - from each VPTN
- Redundancy
- No single point of control
- Resilience to failure

Reliability increases with number of nodes.

Dynamic Virtual Super Peers

- Nodes in a VSP are elected, not preselected
- Continuously, based on stability over time
- Formation of a VSP changes as needed
- Network topology adapts to reflect actual usage



Razavi, Moschoyiannis, Krause. An open digital environment to support business ecosystems. *Peer-to-Peer Networking and Applications* 2(4): 367-397 (2009)

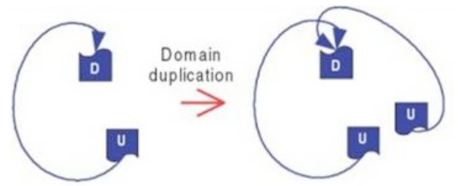
Dynamic topology

- Unstructured network design
  - Dynamicity of local interactions
  - Nodes join and leave the network

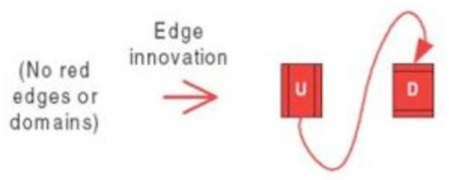
- Biological models --

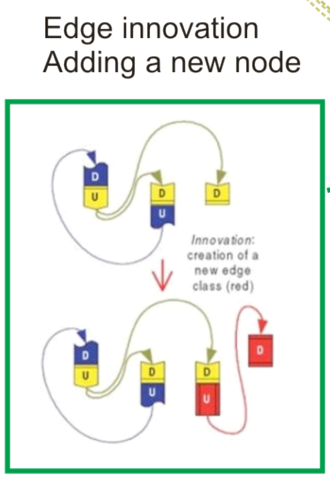
  Growth in molecular networks
  [Gomez & Rhzetsky, 2003, 2005]

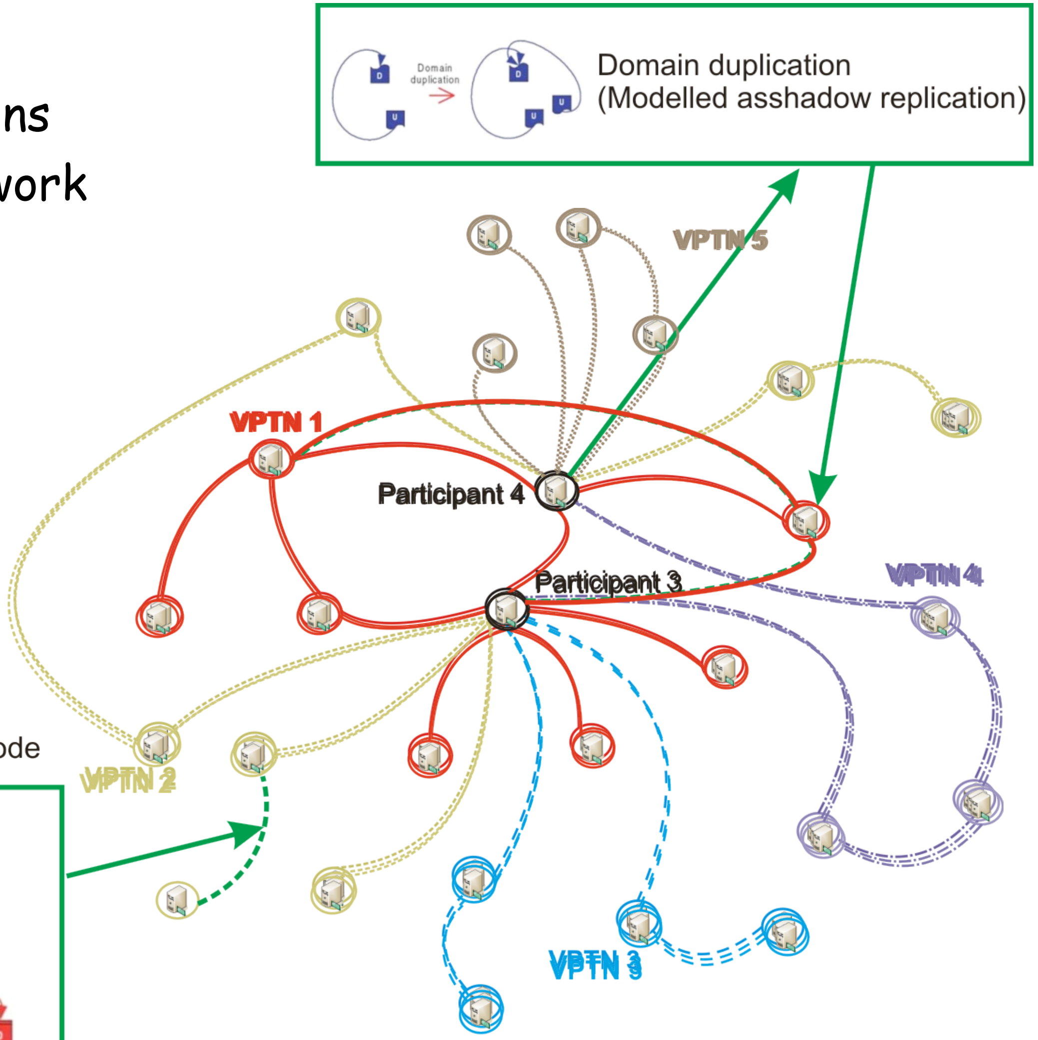
  - domain duplication



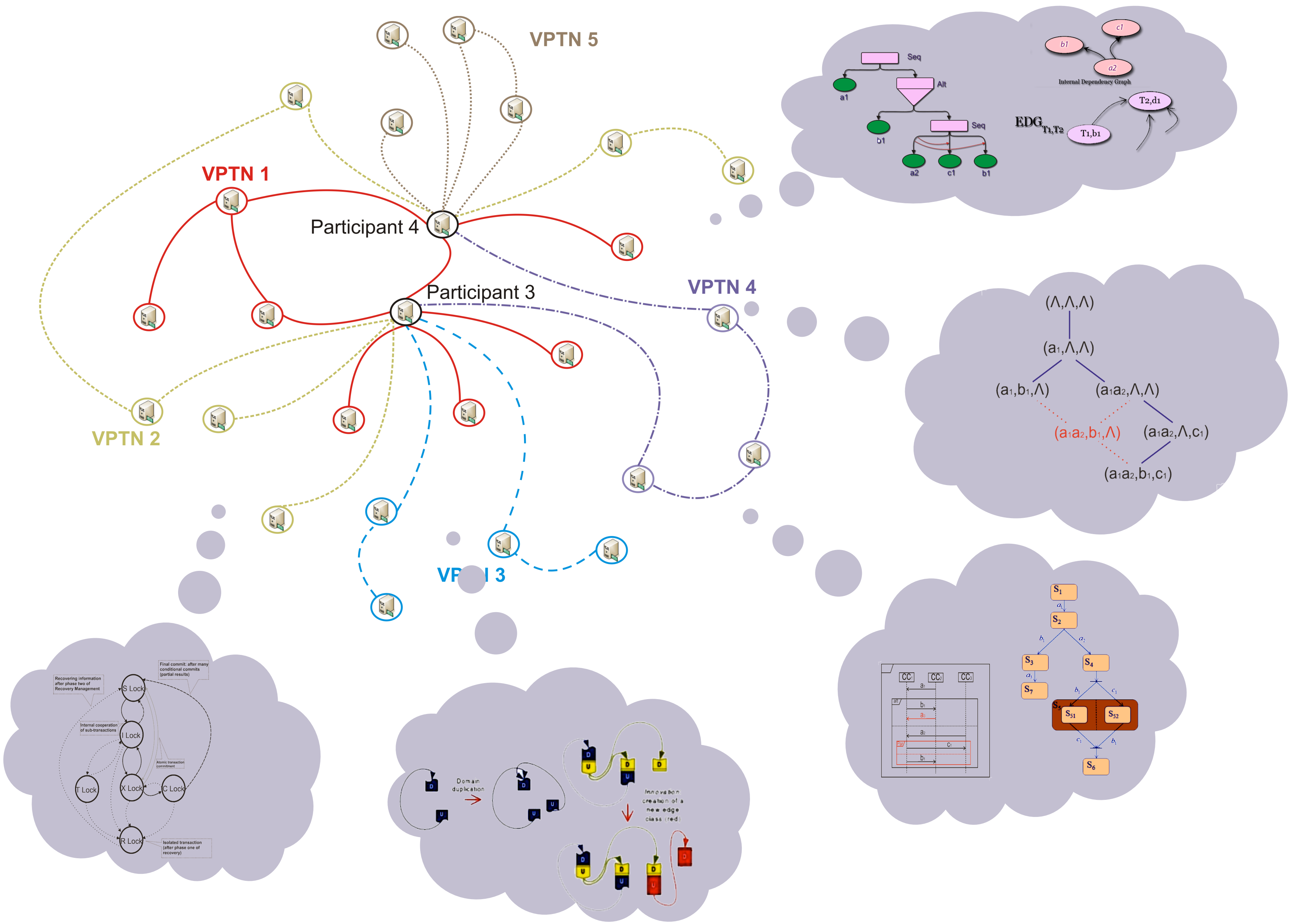
  - edge innovation







Digital environment for open collaboration



Have seen aspects of a self-maintaining environment that evolves to adapt to the complex interactions (B2B, knowledge services) between entities that are organised recursively in smaller and simpler networks.

Have focused on the interactions between the participating nodes.

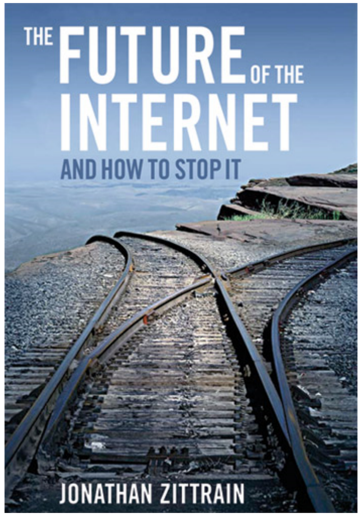
Now, let's take a closer look into a node.

Rules-based approach: what, not how

- Current information systems
  - Tied to a predefined set of business processes
  - Need expert intervention to alter their operation

- Generative information systems
  - Able to satisfy unplanned requests
  - Users empowered to control the logic

- Sterile vs Generative technologies



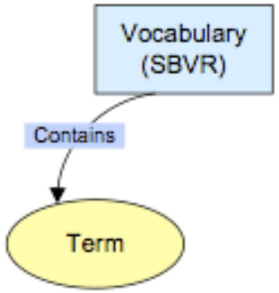
The business rules approach to application development draws on tools / methodologies such as:

Structured Business Vocabulary and Rules (SBVR)
Web architecture (REST over HTTP)
Relational Databases

(*joint work with Alexandros Marinos*)

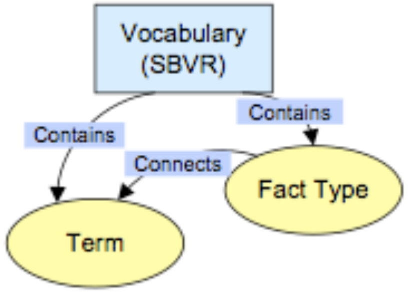
Terms

student
module
course



Terms

student
module
course



Fact Types

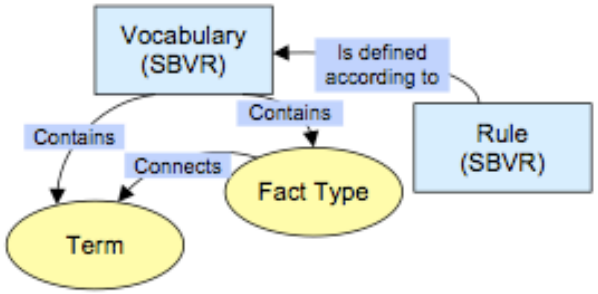
student *is registered for* module

student *is enrolled in* course

module *is available for* course

Terms

student
module
course



Fact Types

student *is registered for* module

student *is enrolled in* course

Module *is available for* course

Rules

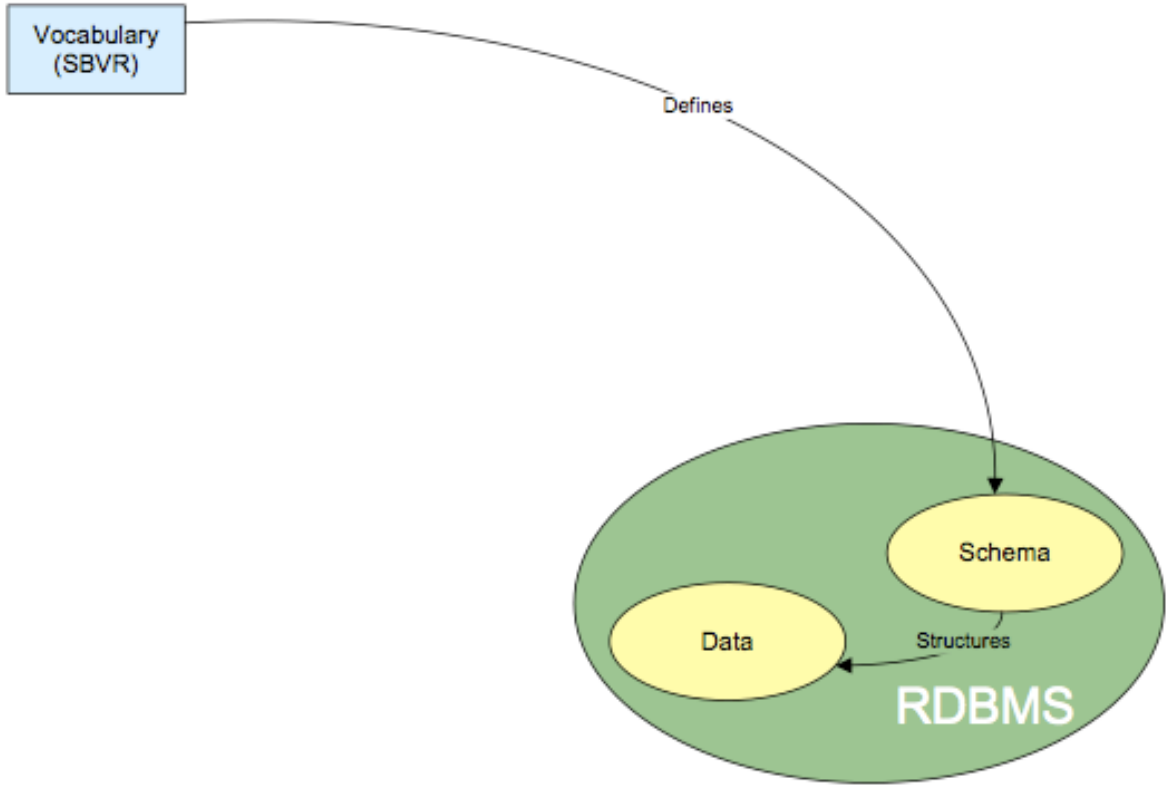
It is necessary that each student *is registered for* at most five courses

It is necessary that each module that a student *is registered for, is available for* a course that the student *is enrolled in.*

Vocabulary

student, module, course

student *is registered for* module
student *is enrolled in* course
module *is available for* course



SBVR to SQL DDL mapping
[Marinos, Moschoyiannis, Krause, 2009]

Vocabulary

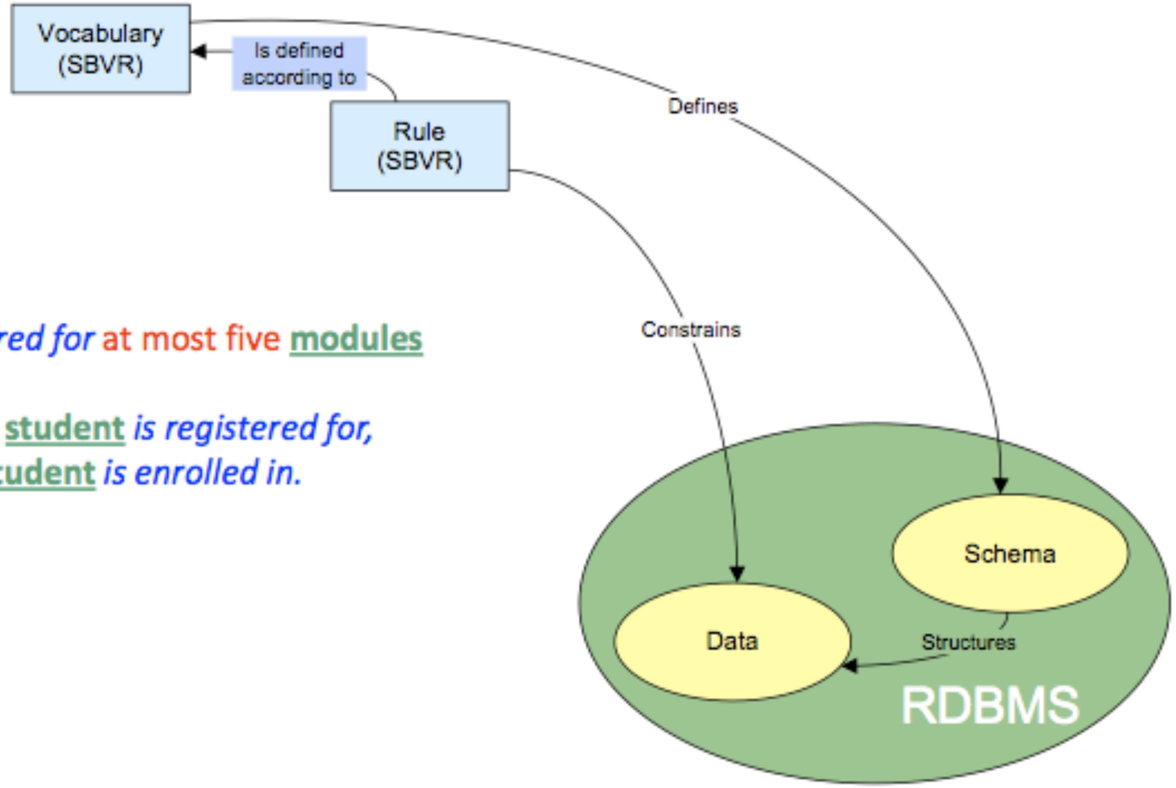
student, module, course

student *is registered for* module
student *is enrolled in* course
module *is available for* course

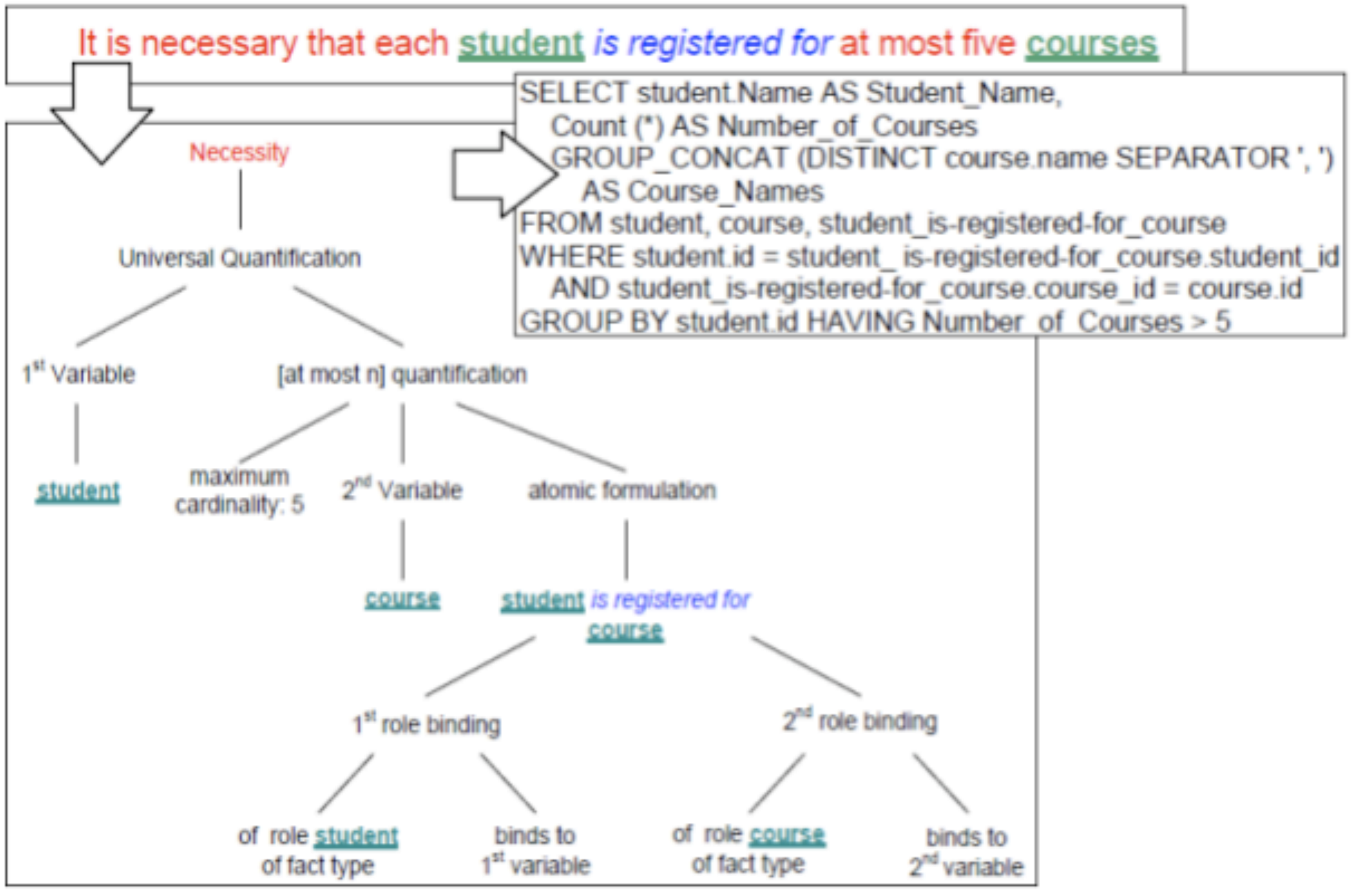
Rules

It is necessary that each student *is registered for* at most five modules

It is necessary that each module that a student *is registered for*, *is available for* a course that the student *is enrolled in.*

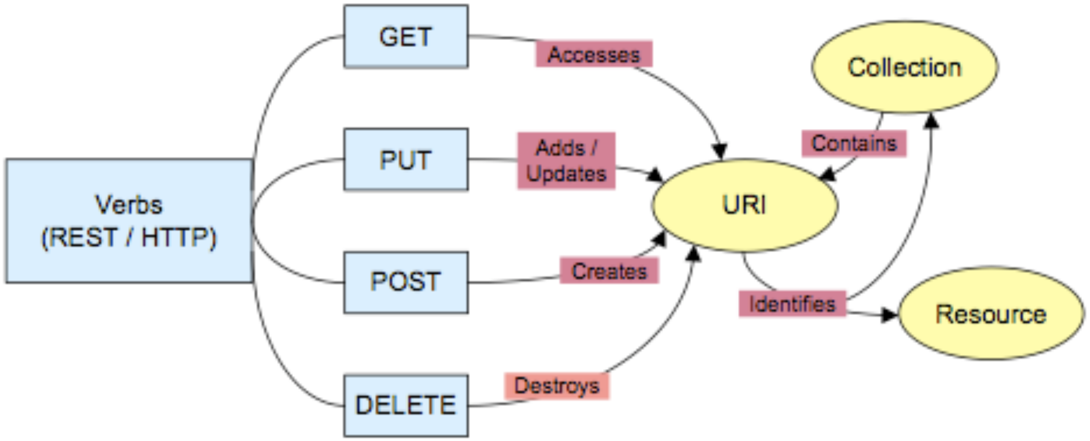


SBVR Rules to SQL Queries

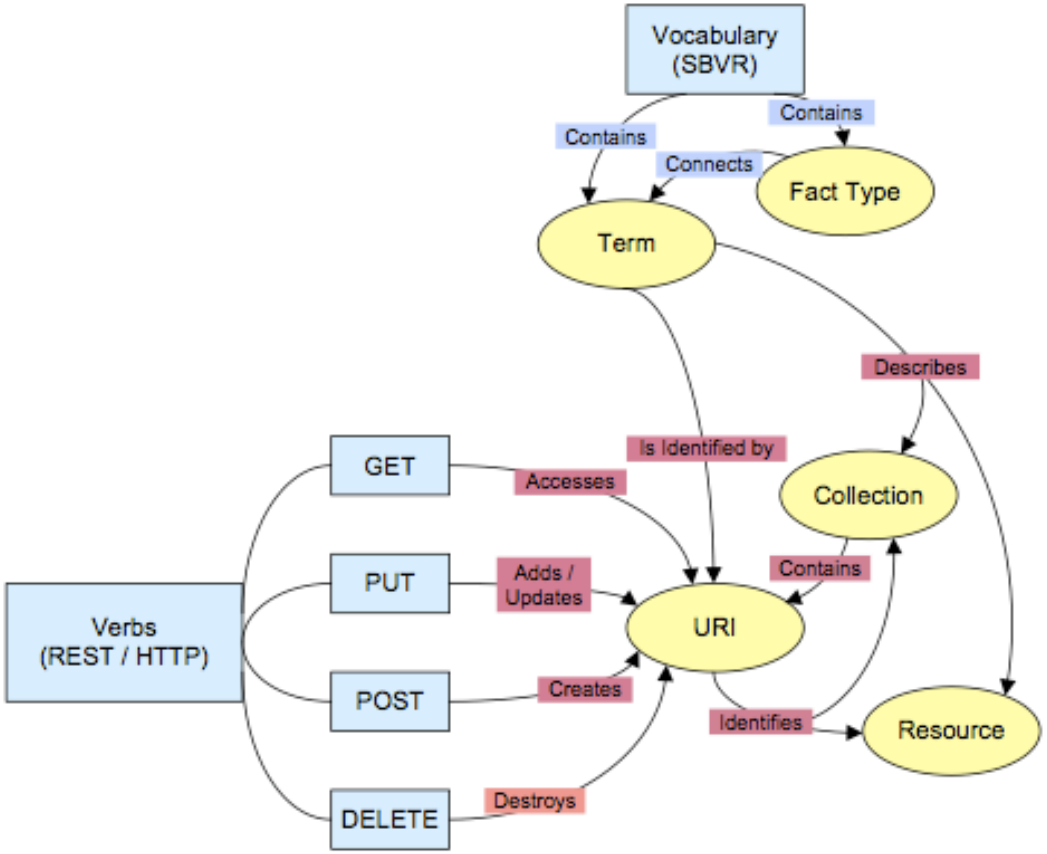


Student_Name	Number_of_Courses	Course_Names
John	6	PY101, MA101, EN121, CS101, AF302, MG102

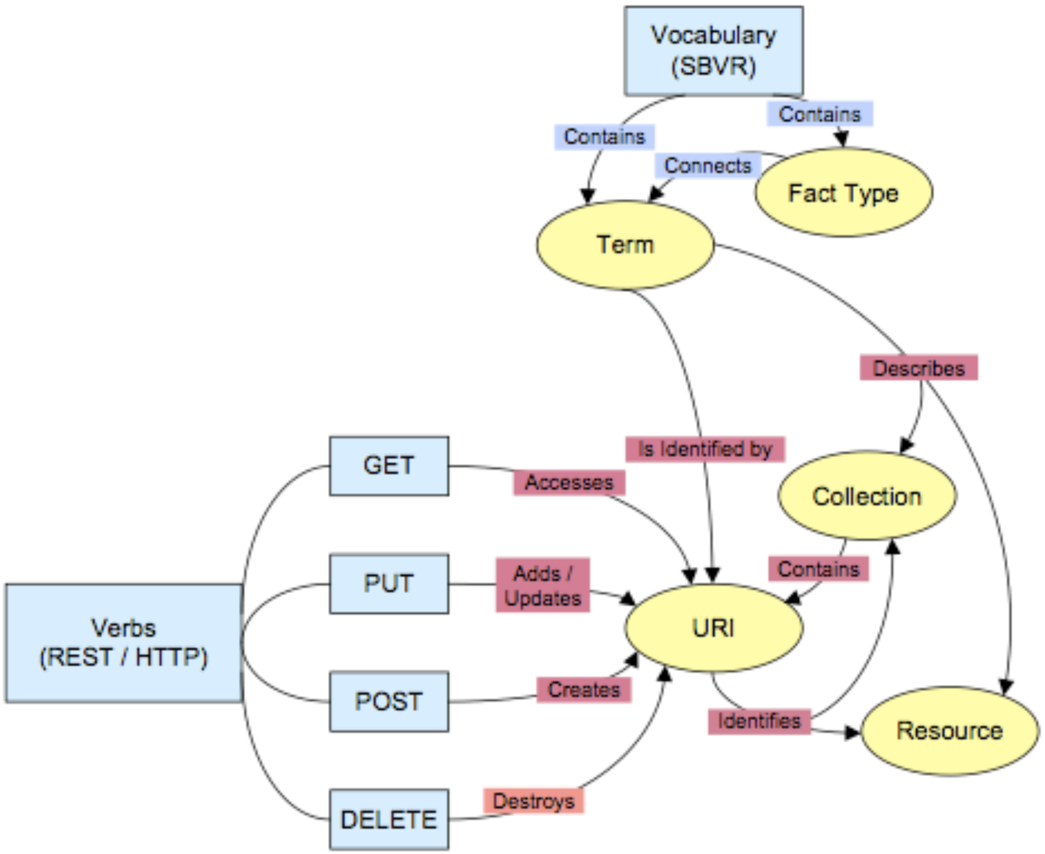
REpresentational State Transfer (REST)



- Important 'things' (nouns) are Resources
  - Addressed through a URI
- Uniform interface (verbs)
  - in HTTP:
    GET, PUT, POST, DELETE
- Verb-noun separation standardises a layer of semantics
- Stateless (loose-coupling)
- Resources should be interconnected via links, to avoid need for out-of-band information



Vocabulary
(SBVR)
Contains
Contains
Connects
Fact Type
Term
Describes
Is Identified by
GET
Accesses
Collection
PUT
Adds /
Updates
Contains
Verbs
(REST / HTTP)
URI
POST
Creates
Identifies
Resource
DELETE
Destroys



Vocabulary

student, course, module, name

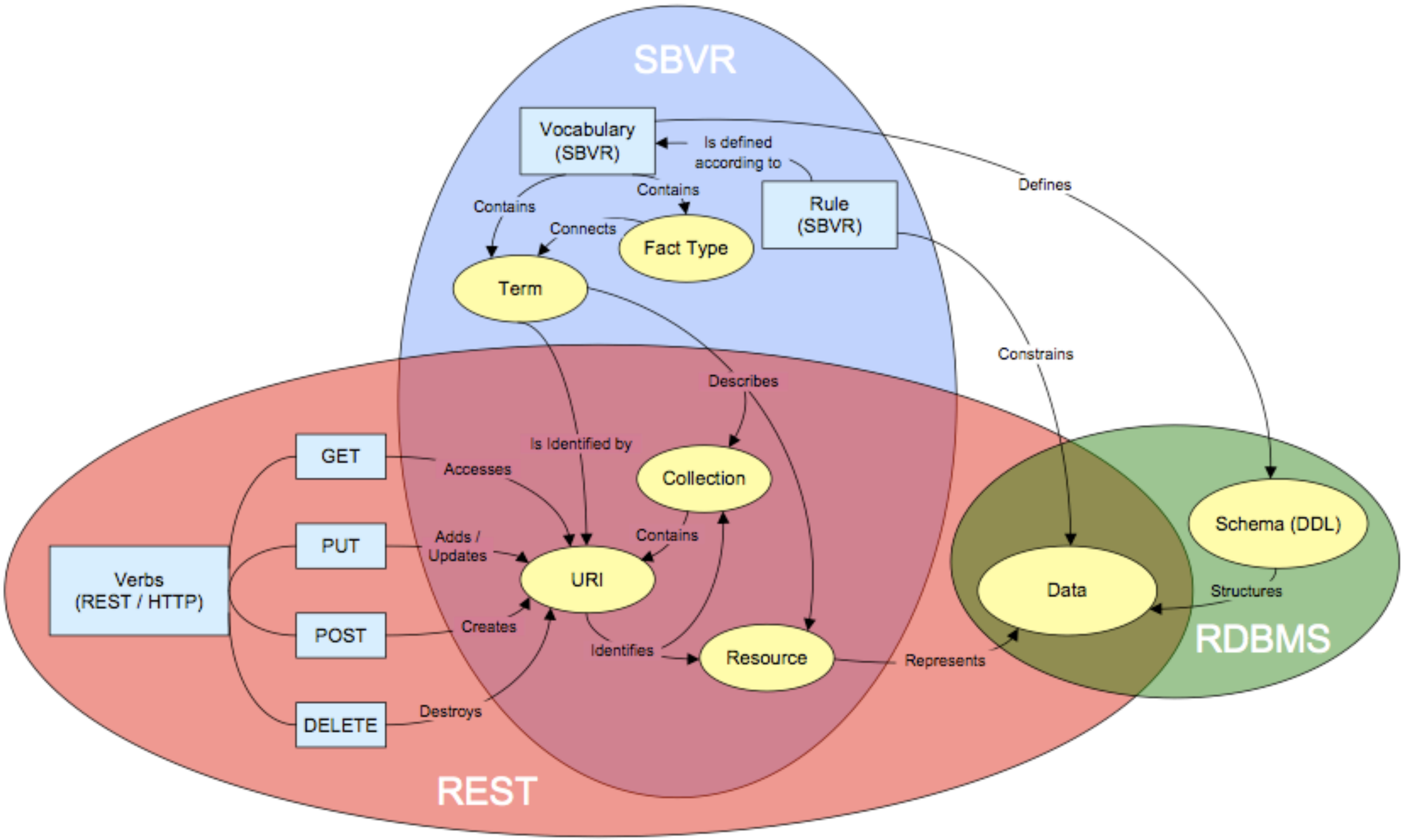
student *is registered for* module
student *is enrolled in* course
module *is available for* course

URIs

collection of students:
http://domain.org/students/

A specific student:
http://domain.org/students/John/

list of courses a student is enrolled in:
http://domain.org/students/John/courses/


SBVR
Vocabulary (SBVR)
Is defined according to
Rule (SBVR)
Defines
Contains
Contains
Connects
Fact Type
Term
Constrains
Describes
Is Identified by
Collection
GET
Accesses
PUT
Adds / Updates
Contains
URI
Verbs (REST / HTTP)
POST
Creates
Identifies
Resource
Represents
DELETE
Destroys
Schema (DDL)
Data
Structures
RDBMS
REST

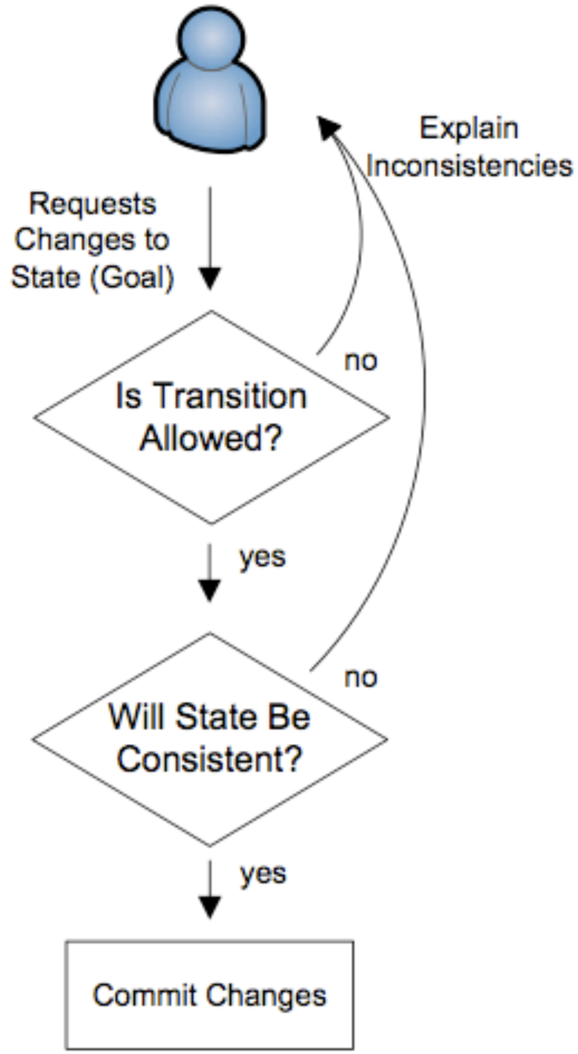
Elementary interactions

	Student		Course		Module	
	Collection	Instance	Collection	Instance	Collection	Instance
GET	+	+	+	+	+	+
PUT		+		+		+
POST	+		+		+	
DELETE		+		+		+

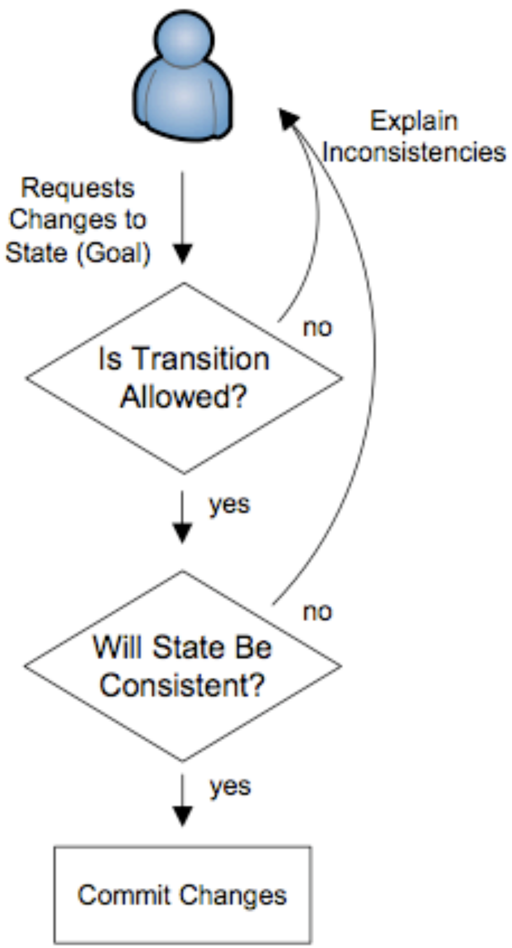
- GET http://domain.org/Students/
  - Collection of students is returned

- DELETE http://domain.org/Student/John
  - Student John is deleted

Process-like behaviour



Process-like behaviour



User:
POST <en101>
http://domain.org/students/John/courses/

System:
403 Forbidden

It is necessary that each **student** *is registered for* at most five **courses**

Student_Name	Number_of_Courses	Names_of_Courses
John	6	PY101, MA101, EN121, CS101, AF302, MG102

User:
[Start Transaction]
DELETE
http://domain.org/students/John/courses/ma101

POST <en101>
http://domain.org/students/John/courses/
[End Transaction]

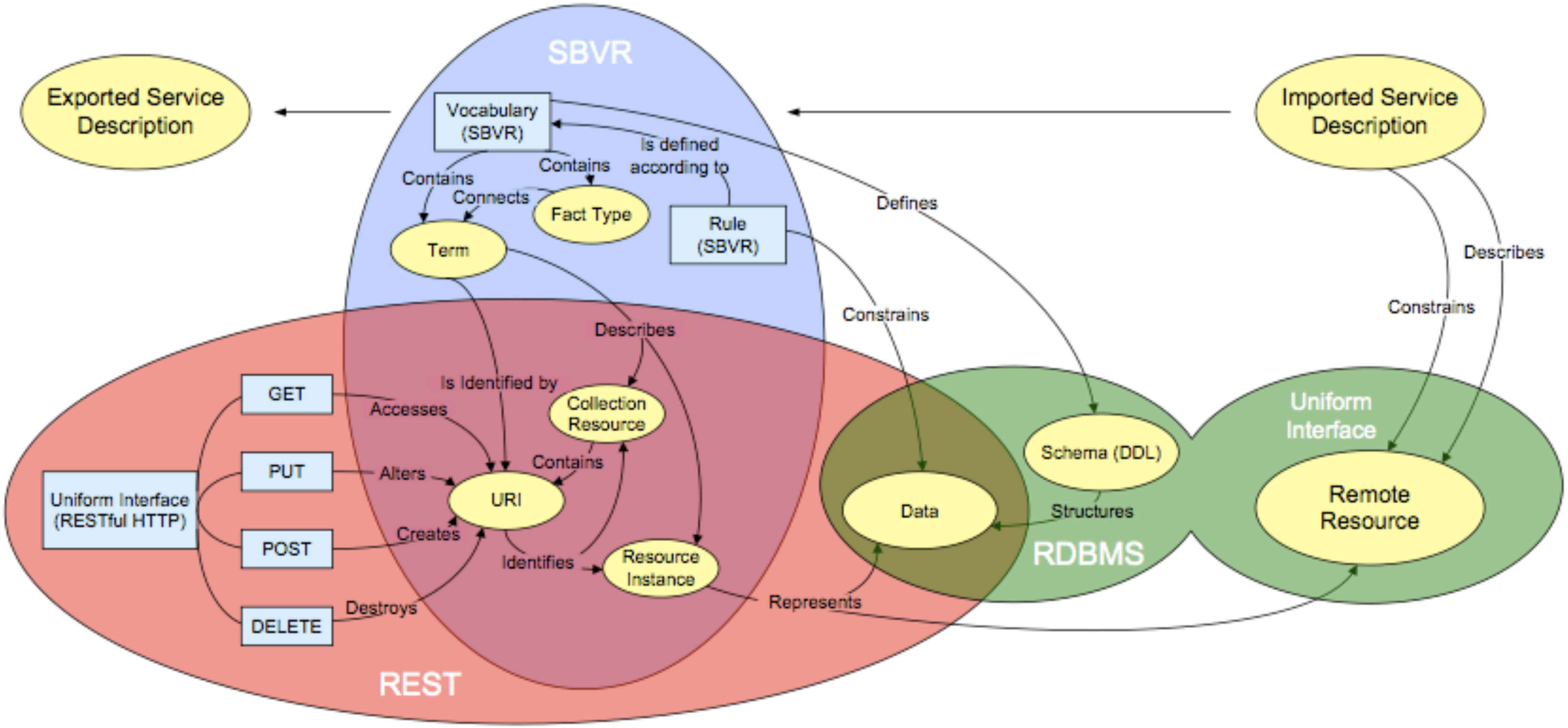
System:
200 OK

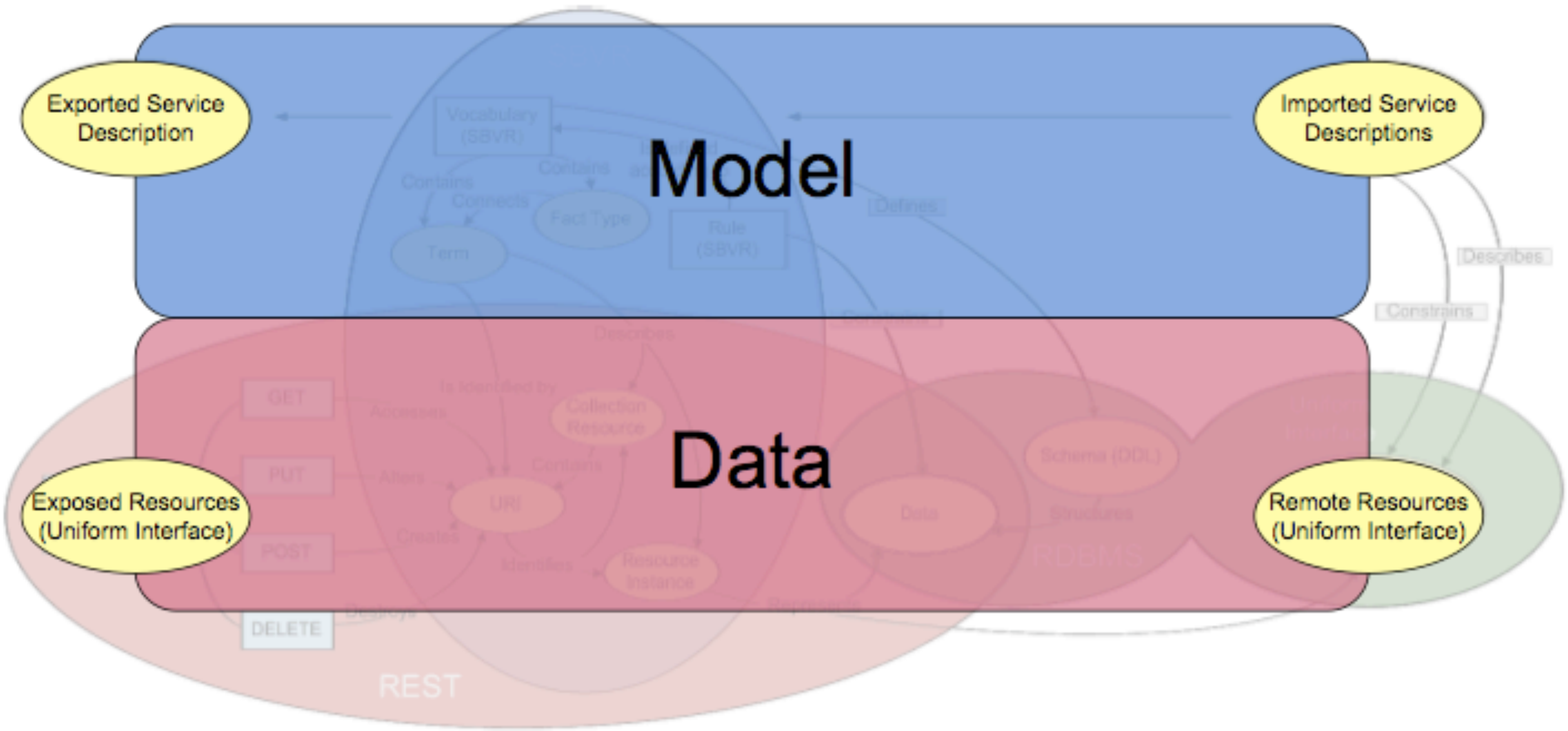
SBVR for resource description

```
<?xml version="1.0" encoding="UTF-8"?>
<student>
          <id>3465</id>
          <firstname>John</lastname>
          <lastname>Smith</lastname>
          <is-under-probation value="false" />

          <link rel="is-enrolled-in_modules"
              href="http://domain.org/school/student/3465/is-enrolled-
              in/modules" />

          <link rel="is-registered-for_course"
              href="http://domain.org/school/student/3465/is-
              registered-for/courses" />

          <link rel="is_marked_with-grade-for-course"
              href="http://domain.org/school/student/3465/is-marked-
              with/grade/for/courses" />
</student>
```

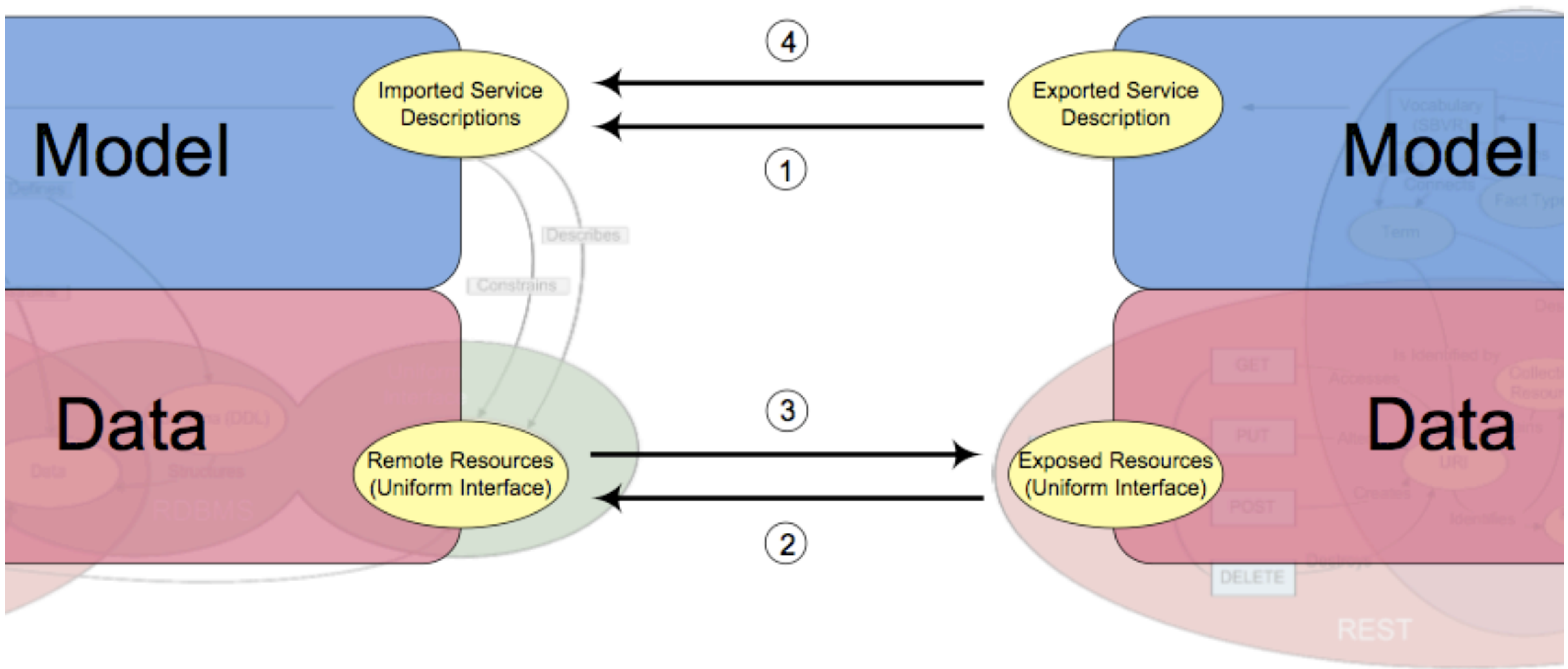
Terms	Fact Types	Rules
Student Module Course Grade A or B or C or D or F First name Last name	Student *is under probation* Student *is registered for* course Student *is enrolled in* module Student *has* first name Student *has* last name Student *is marked with* grade *for* course Module *is available for* course	It is necessary that each student *is registered for* at most five courses. It is necessary that each module that a student *is registered for is available for* a course that the student *is enrolled in*. It is necessary that each student that *is under probation is registered for* at most three courses.

Future directions

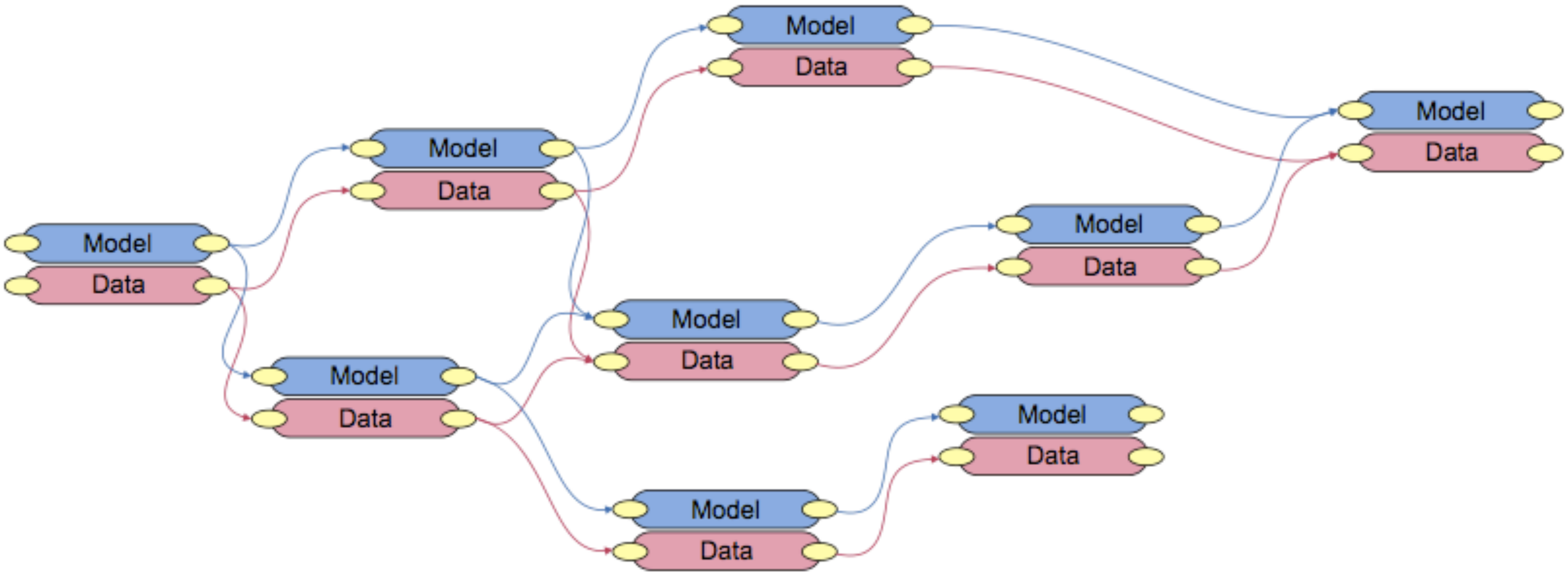


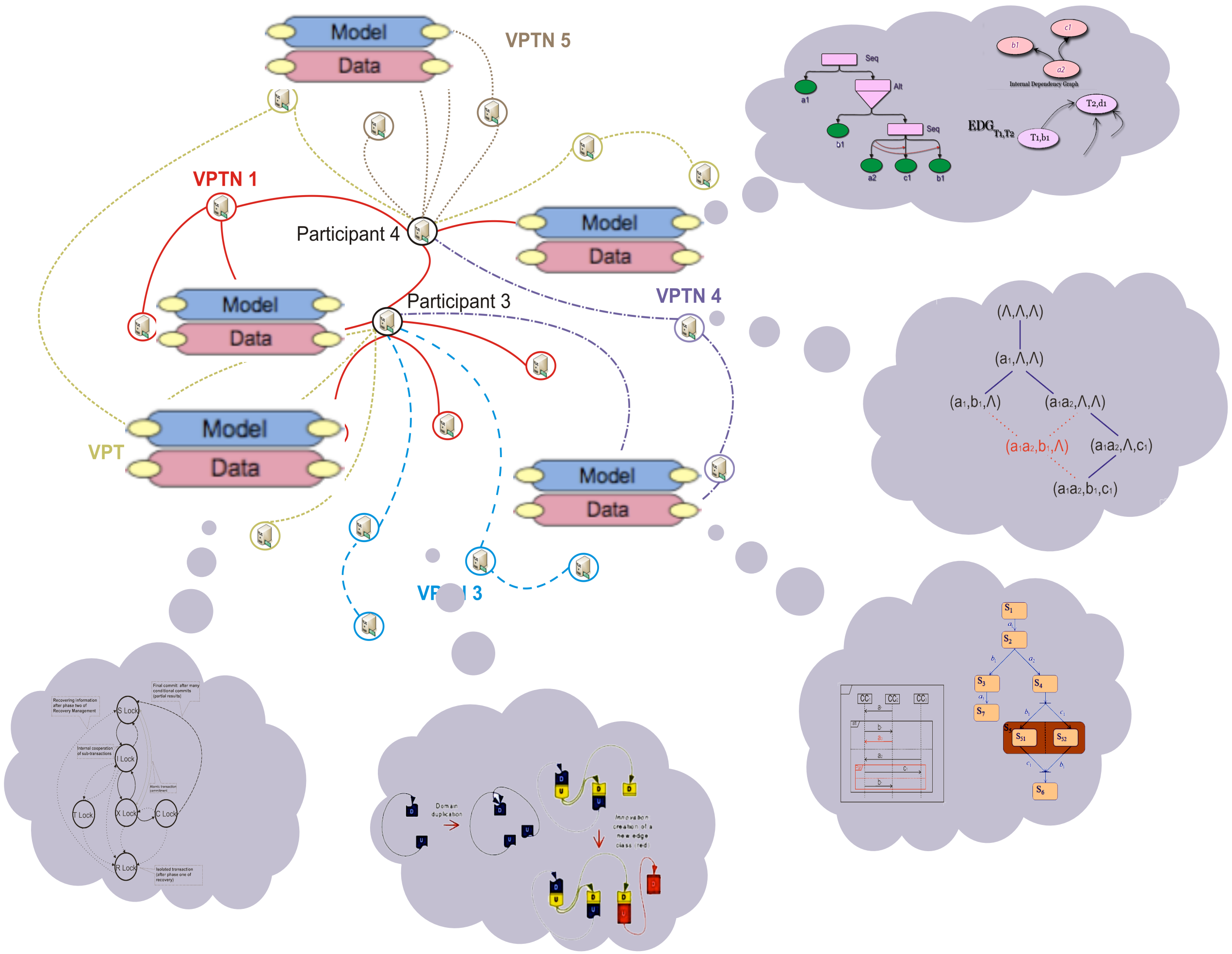


Exported Service Description
Imported Service Descriptions
Model
Data
Exposed Resources (Uniform Interface)
Remote Resources (Uniform Interface)
DELETE
REST



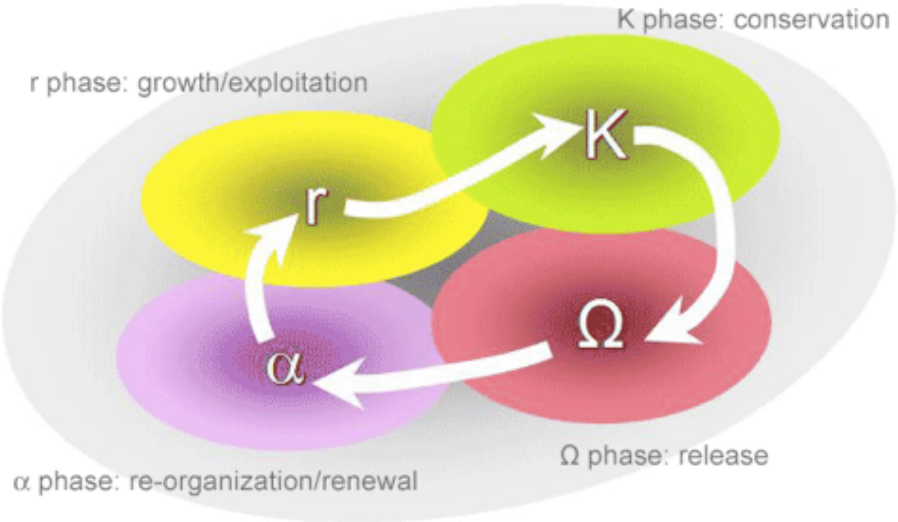
Model
Imported Service Descriptions
4
1
Exported Service Description
Model
Data
Remote Resources (Uniform Interface)
3
2
Exposed Resources (Uniform Interface)
Data


Model
Data
Model
Data
Model
Data
Model
Data
Model
Data
Model
Data
Model
Data
Model
Data
Model
Data

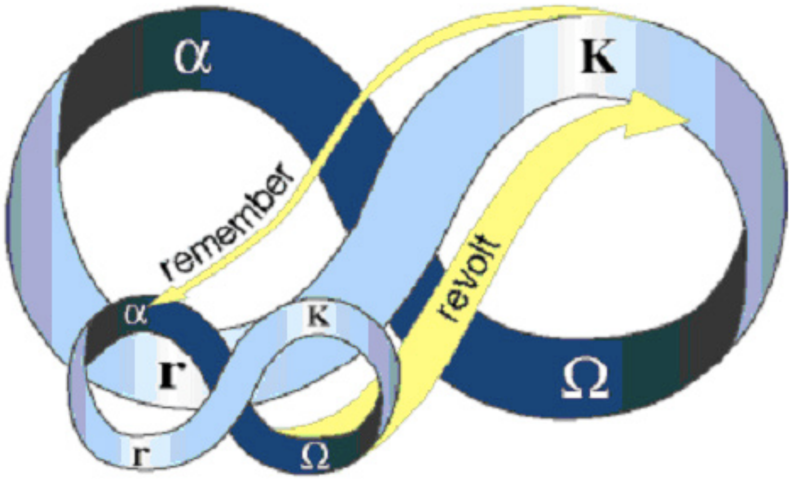


Model
Data
VPTN 5
VPTN 1
Participant 4
Participant 3
VPTN 4
VPT
VPTN 3
(Λ,Λ,Λ)
(a₁,Λ,Λ)
(a₁,b₁,Λ)
(a₁a₂,Λ,Λ)
(a₁a₂,b₁,Λ)
(a₁a₂,Λ,c₁)
(a₁a₂,b₁,c₁)
Seq
Alt
Internal Dependency Graph
EDG

Dynamics of ecosystems – complex adaptive cycles

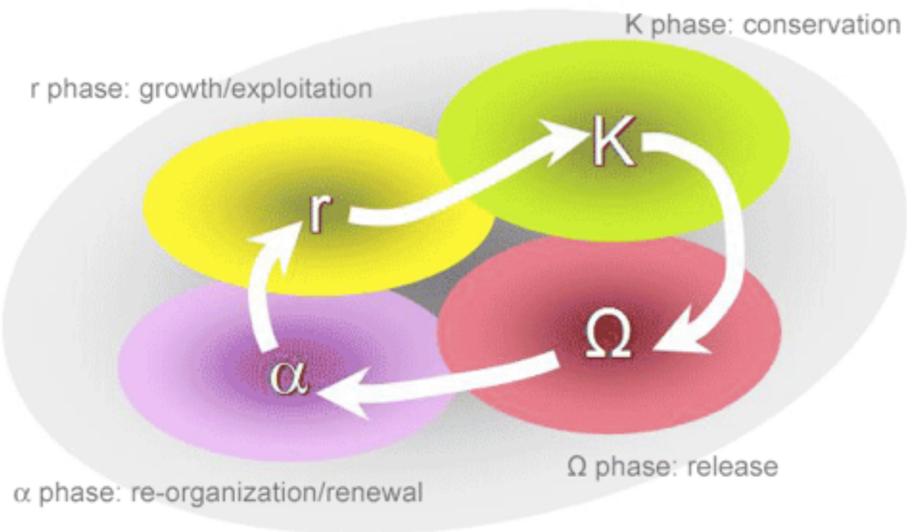


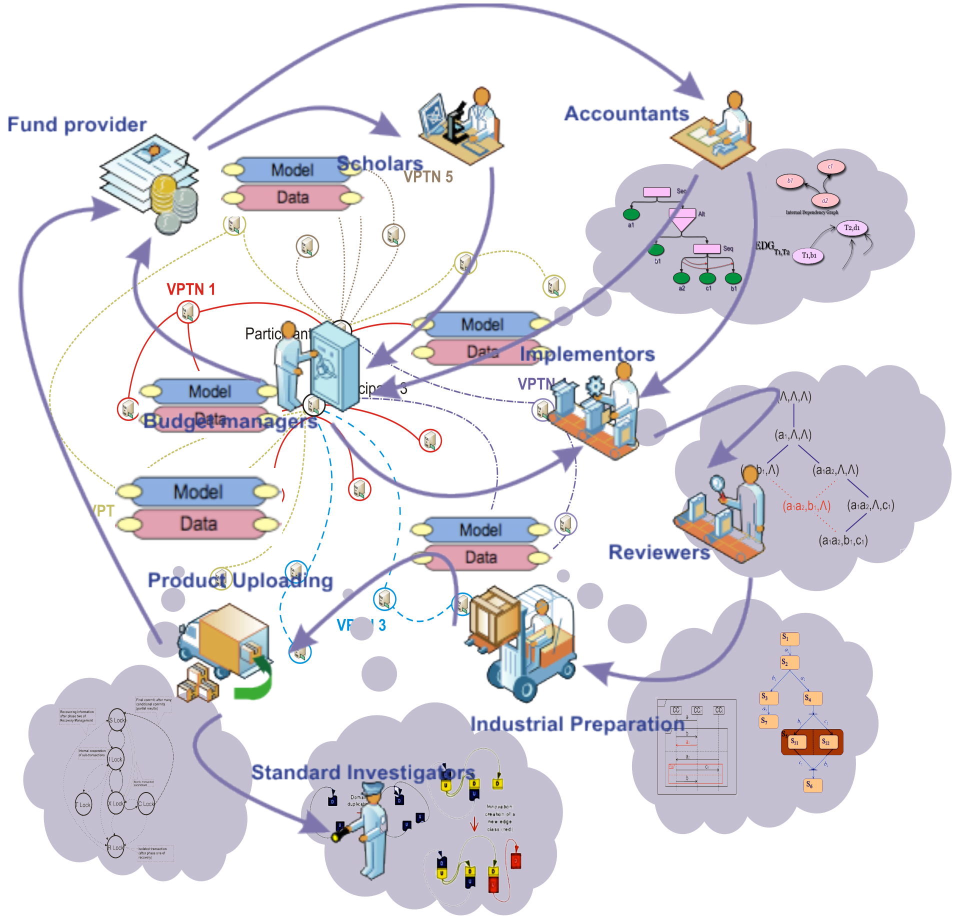
[Holling, Gunderson, 2002, 2006]



- Processes of growth (r) and conservation (K)
  - slow, incremental
  - connectedness, stability increase (skills, networks of relationships)
- ..but also of destruction (Omega) and re-organisation (alpha)
  - rapid, leading to renewal
  - adaptive capacity, opportunities for innovation, new configurations
- Connected adaptive cycles..
  - non-linear, multi-scale at each level
  - properties stabilised or destabilised across levels

Research coordinates





- Emergence, immergence
- Multi-level, multi-player

  ...

- Resilience, sustainability

=> Digital environment to support complex socio-economic systems

Research coordinates

- Incorporate (relevant) concepts from digital ecosystems
- Analytical tools and methods for reasoning / prediction
  - goal-oriented req engineering [e.g. Letier, Kramer, Magee, 2008]
  - model-checking techniques [e.g. Kwiatkowska, Norman, Parker, 2004]
  - concurrency, non-determinism, alternative scenarios
- Distributed aspects are paramount in such complex systems
  - services made available and consumed (*transient*)
  - organised in an architecture that mirrors the web

OPAALS

ERIE
Evolution and Resilience of Industrial Ecosystems

SEVENTH FRAMEWORK PROGRAMME

Sixth Framework Programme

EPSRC
Pioneering research and skills

Thank you for your attention !