



Universidad  
Rey Juan Carlos

Formal Models and Technologies for Emerging Applications S2018/TCS-4314

# FORTE-CM Kick-Off meeting

Kybele Research Group | Universidad Rey Juan Carlos



① Kybele at a glance

② INNoVaServ

③ SmaC

## Full-time PhD

Esperanza Marcos (LEAD)	(CU)
Juan M. Vara	(PTU)
M <sup>a</sup> Valeria de Castro	(PTU)
Marcos López Sáenz	(PTU)
David Granada	(PV)
Fco. Soltero	(PV)

## Full-time PhD Students

Fco. Javier Pérez	(PV)
Maricela Salgado	(PV)
Cristián Gómez	(PI Predoc. CAM)
Daniela Fabersani	(PI FORTE-CM)

## Part-time PhD Students

Roberto Hens  
Juan Canela Vicente

## Research Technicians

Alberto Fernández	(PI MINECO)
XXX	(TI CAM)

## SOFTWARE ENGINEERING AND INFORMATION MANAGEMENT

### Industrial Software Development:

Technologies and Methods

Model Driven Engineering

Software Modernization

Product Lines

Software Factories

Data Management (DW, XML, Grid...)

Ontologies and Models in IS Development

### Information Systems Security

Risk Management Methodologies

Security Patterns

### Philosophical Foundations of IS engineering

Research Methods in Soft. Engineering

Gender in Software Engineering

### Software and Process Quality

Software Process Improvement

Agile Methods

Value – Based Software Engineering

Software Quality Assurance

### Service Science and Engineering

Service Arquitectures

Service developement Methods

Business Processess

Web Engineering

### Human Aspects of SE

Coaching

Team Management

Emotional Intelligence



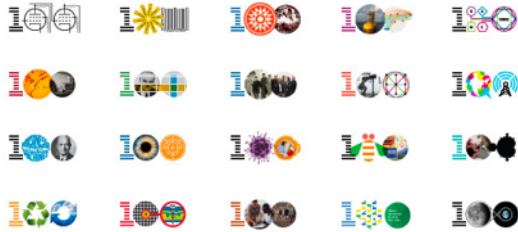
- Software Engineering
  - Model-Driven Engineering
  - Agile Methodologies
  - Human aspects of SE
  - Software Quality
- Services Science Management and Engineering
  - Business and Business Process Modeling
  - Service Design
  - Services Engineering
  - Service Quality

- Software Engineering
  - Model-Driven Engineering
  - Agile Methodologies
  - Human aspects of SE
  - Software Quality
- Services Science Management and Engineering
  - Business and Business Process Modeling
  - Service Design
  - Services Engineering
  - Service Quality

① Kybele at a glance

② INNoVaServ

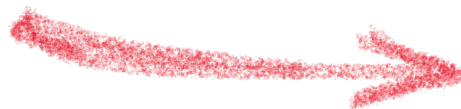
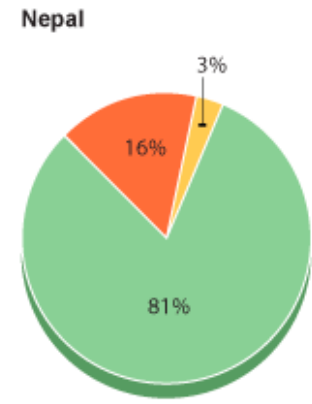
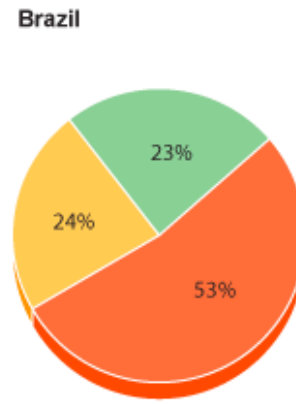
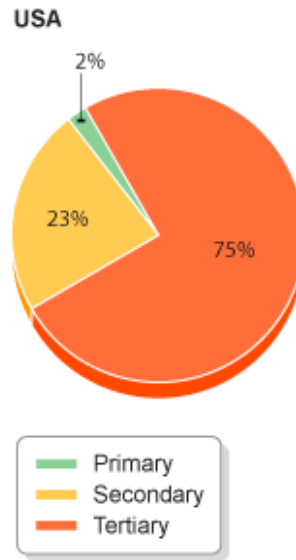
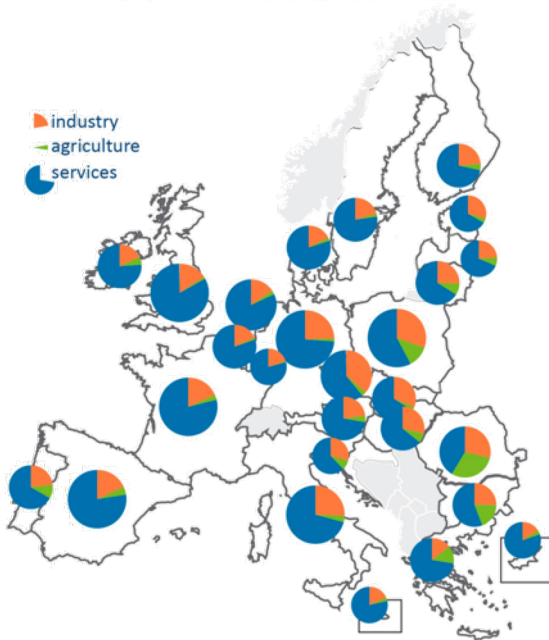
③ SmaC



## The Invention of Service Science

Just as IBM in the 1940s helped create the academic discipline of computer science, so the company is again extending scientific rigor to key emerging dimensions of a changing world. With the world's economy shifting from manufacturing to services, Service Science, Management and Engineering (SSME) introduces an important new field of study to enable deeper understanding of how this shift manifests itself in particular organizations and across business and society. Since 2003, IBM has worked with 450 university faculties in 54 countries, as well as governments and industry leaders, to build SSME curricula.

1. Employment in industry, agriculture and services - 2013




**GRADO Y MÁSTER INGENIERÍA DE SERVICIOS**

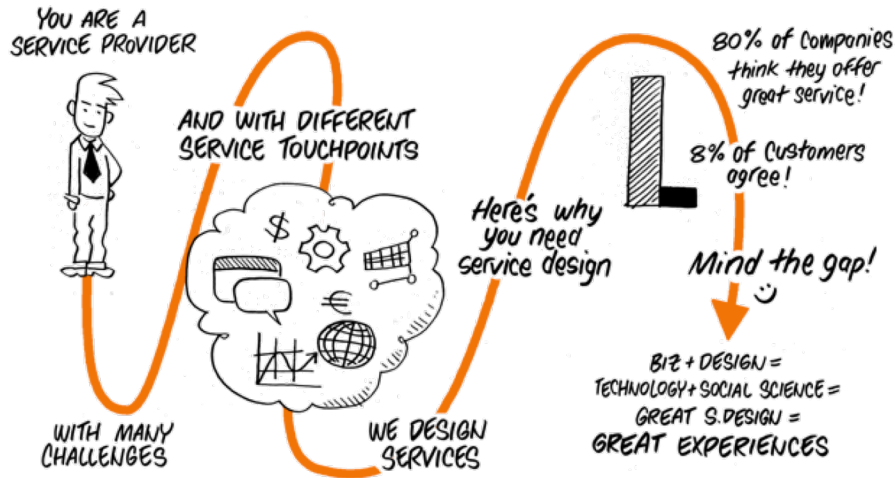
- Salidas profesionales a nivel internacional.
- Para dirigir y gestionar empresas y departamentos de organizaciones de servicios (turismo, banca, IT, facility o cualquier otro sector).
- Profesorado interdisciplinar con amplia experiencia y colaboración de empresas con proyección internacional en diferentes sectores (IBM, EULEN, Melia, BBVA, etc).

**BIENVENIDO A LA COMUNIDAD**

**¿QUÉ ES LA INGENIERÍA DE SERVICIOS?**

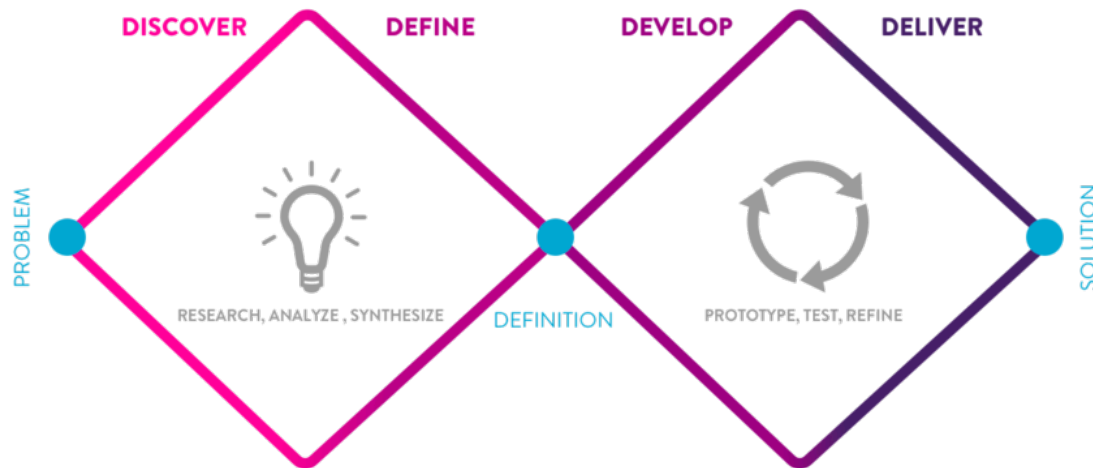
**POR QUÉ ESTUDIAR INGENIERÍA DE SERVICIOS**

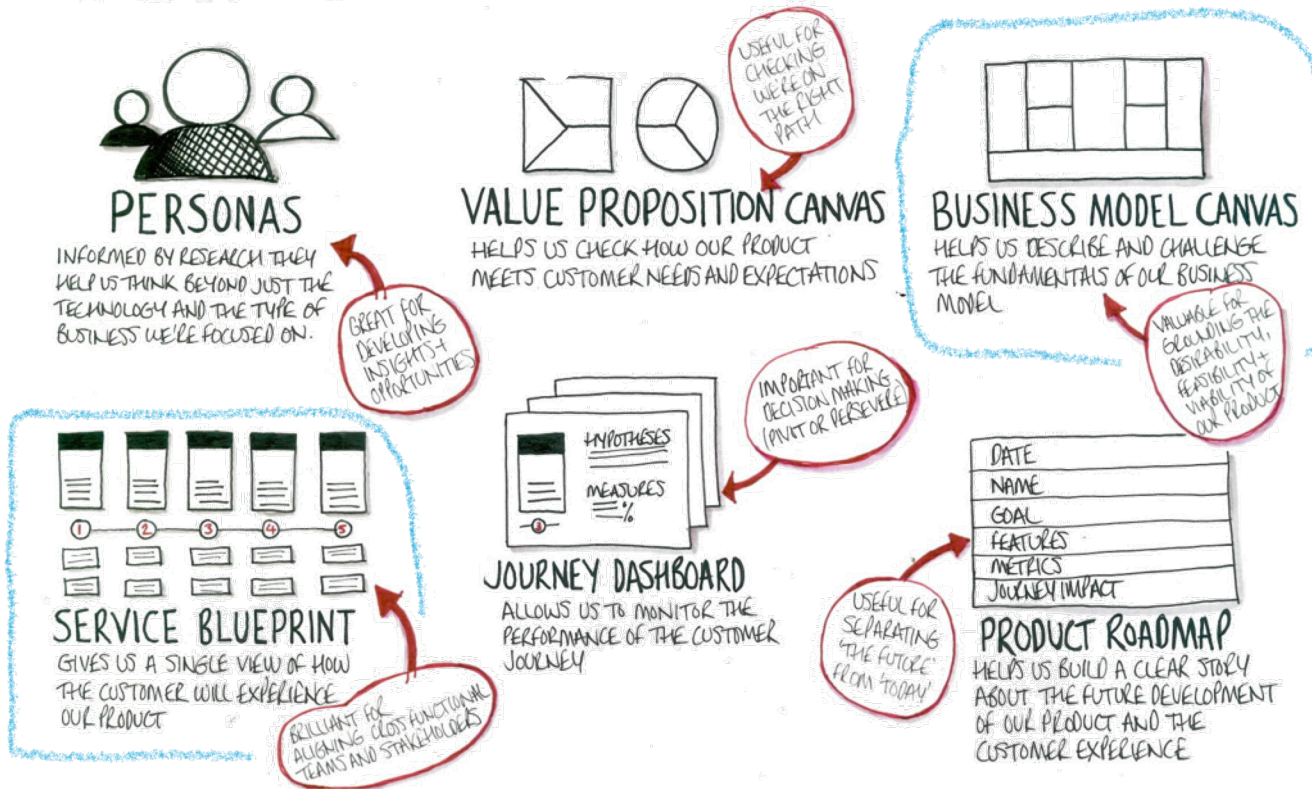
**NEGOCIO**  
**PERSONAS**  
**TECNOLOGÍA**



Service design is the activity of planning and organizing a business's resources (people, props, and processes) in order to

- (1) directly improve the employee's experience, and
- (2) indirectly, the customer's experience.





Process-based nature

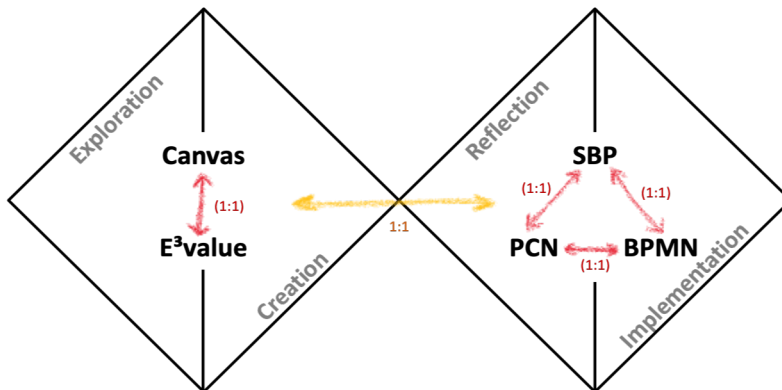
Value  
co-creation

Holistic  
point of view





Business and Business Process Modeling toolkit for Service Design



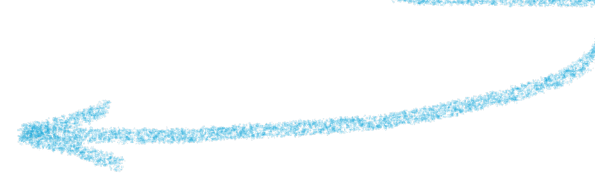
**Canvas**

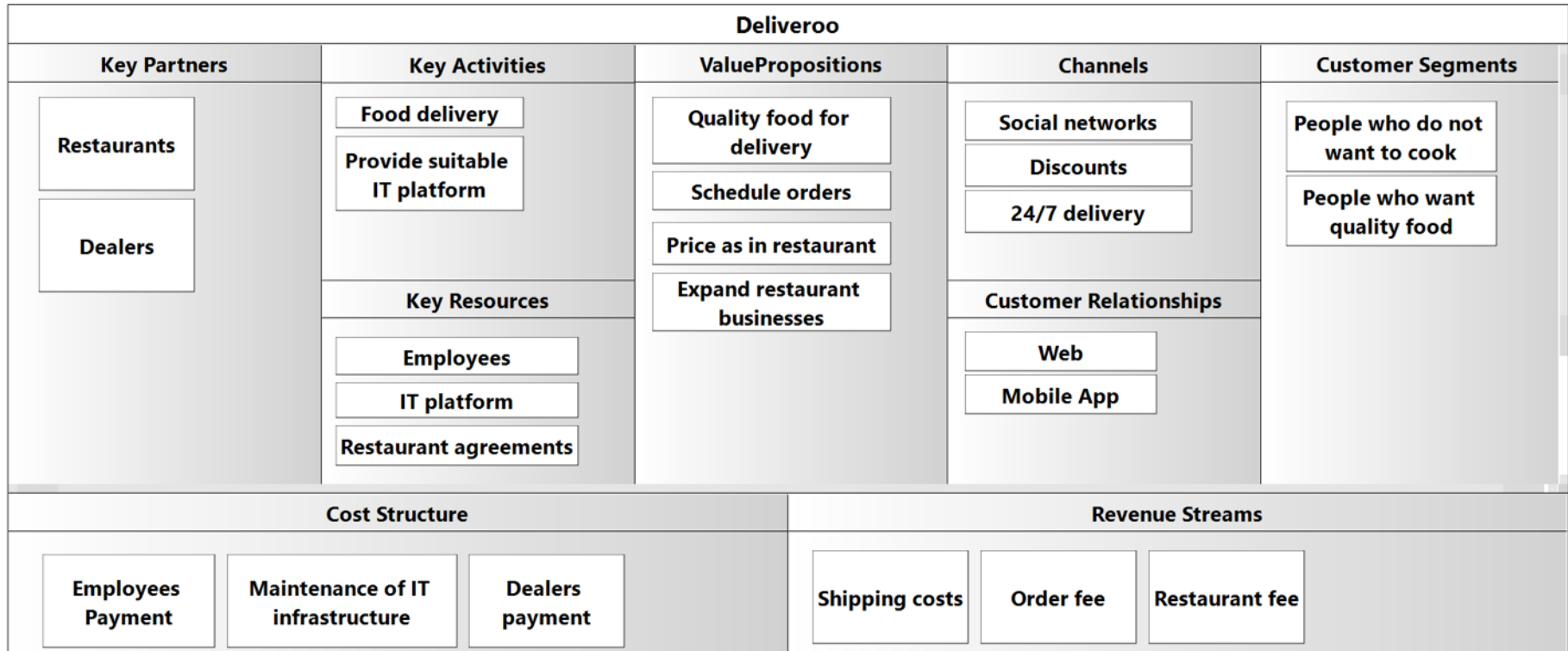
**e<sup>3</sup>value**

**PCN**

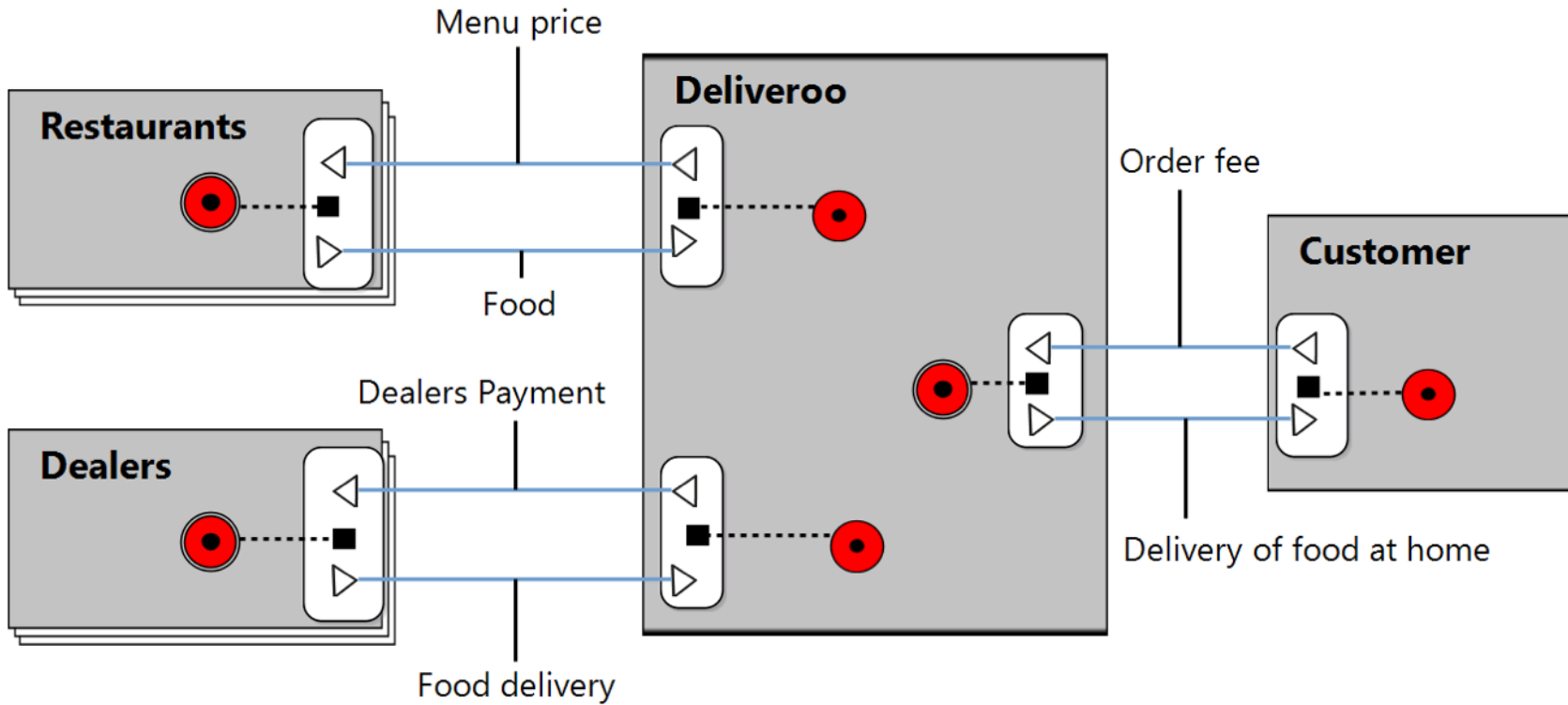
**Service  
Blueprint**

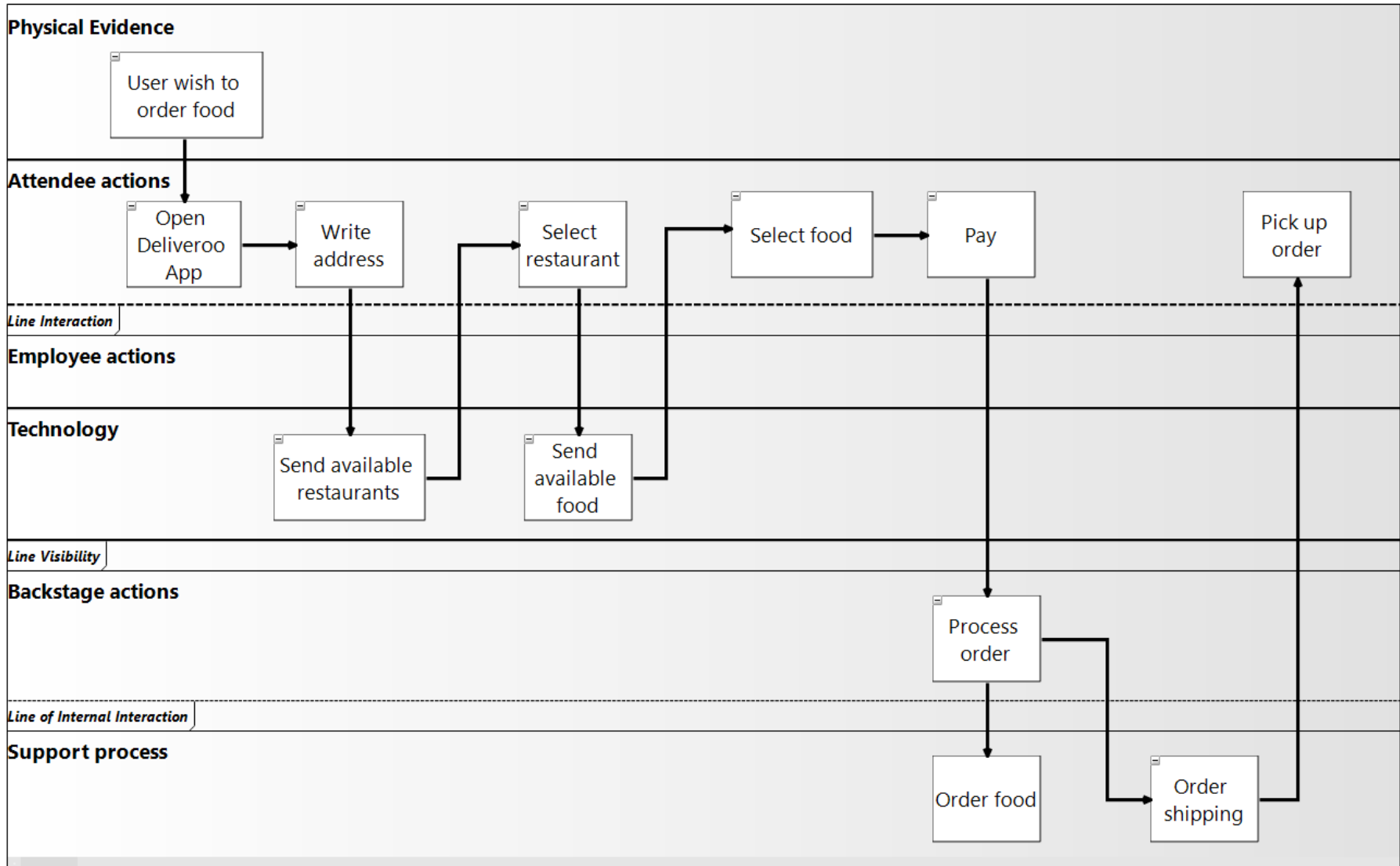
**BPMN**

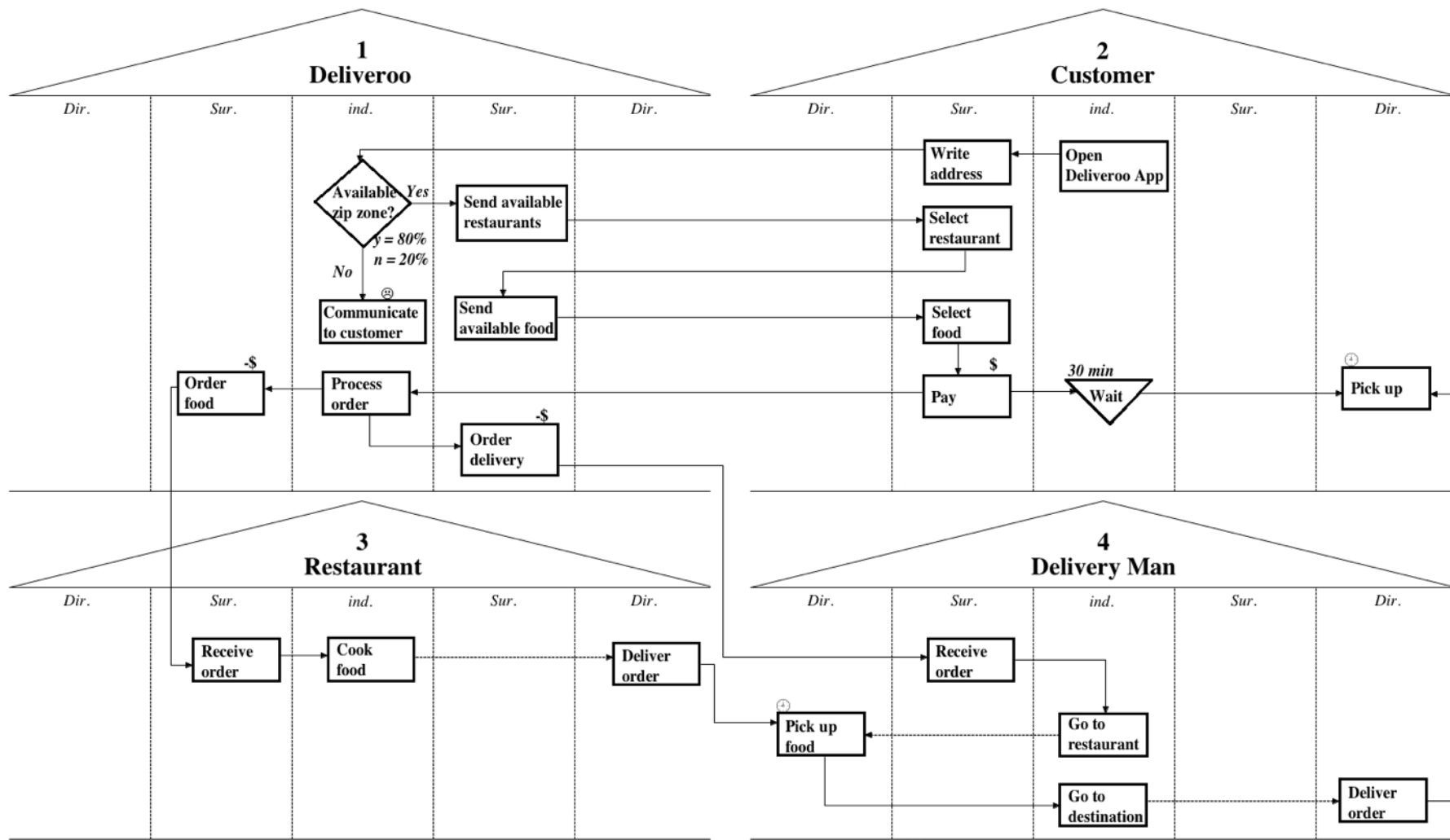


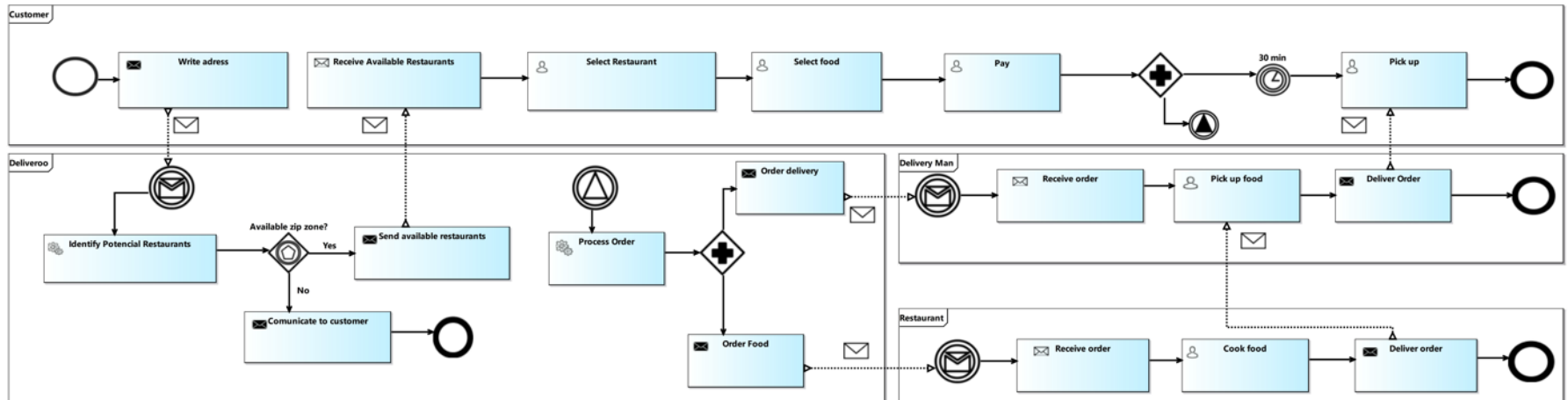




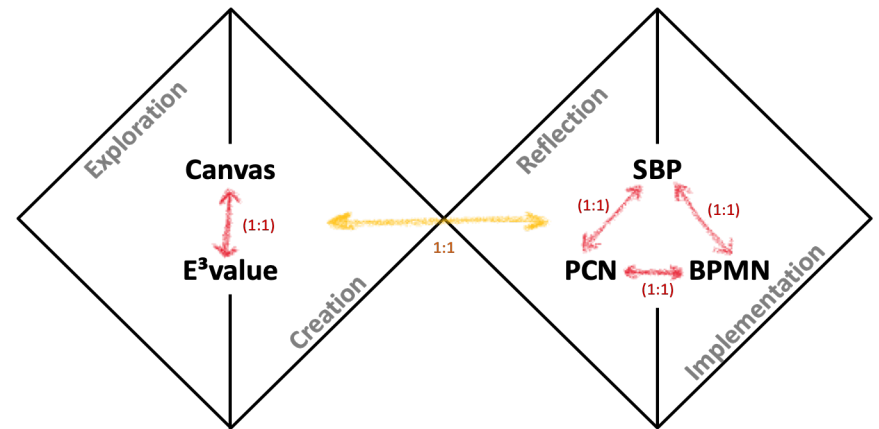




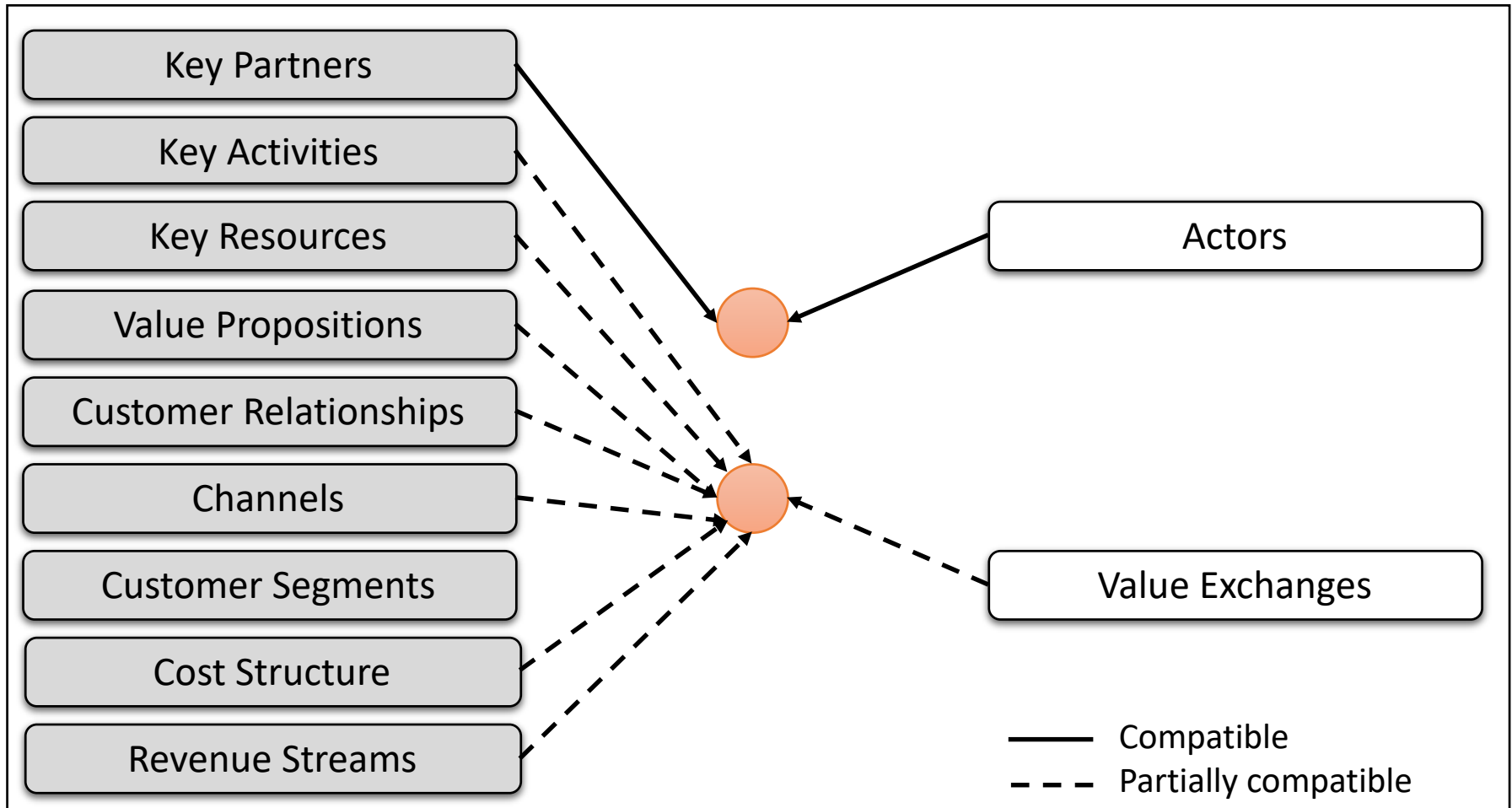


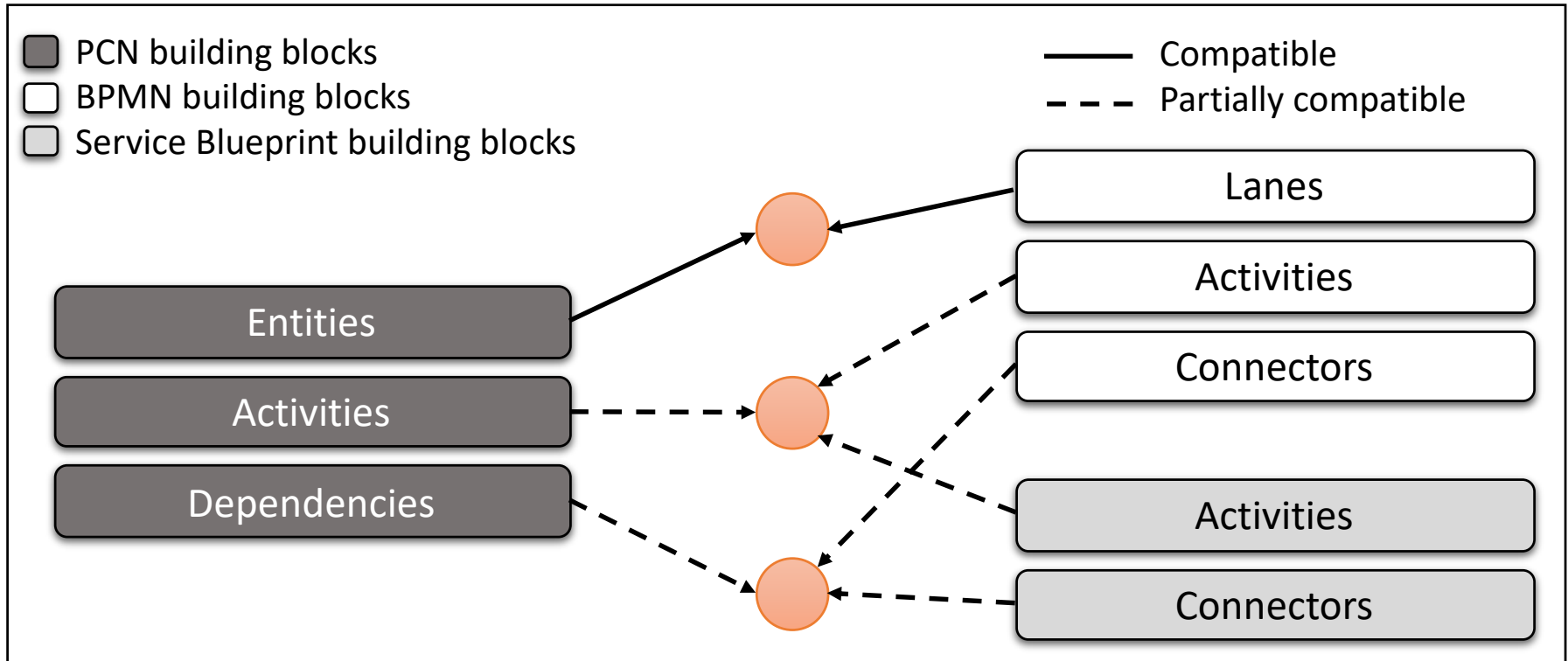


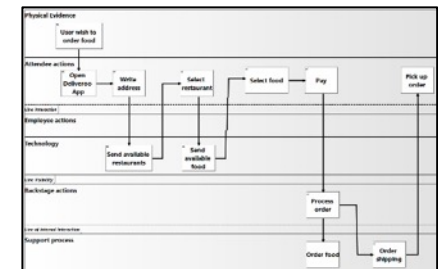
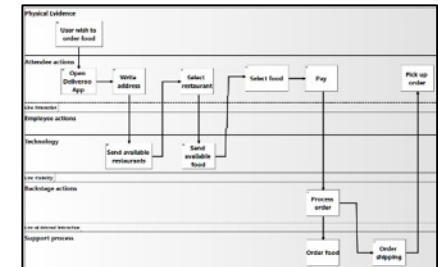
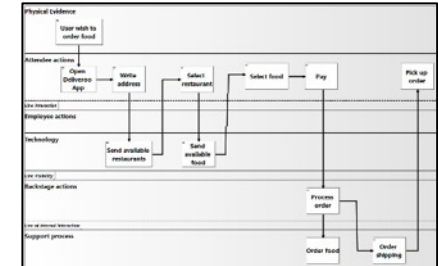
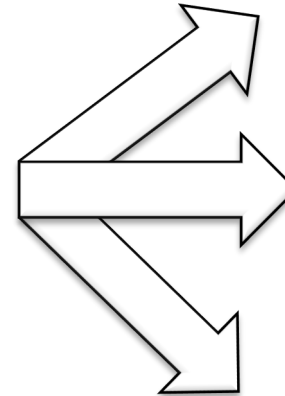
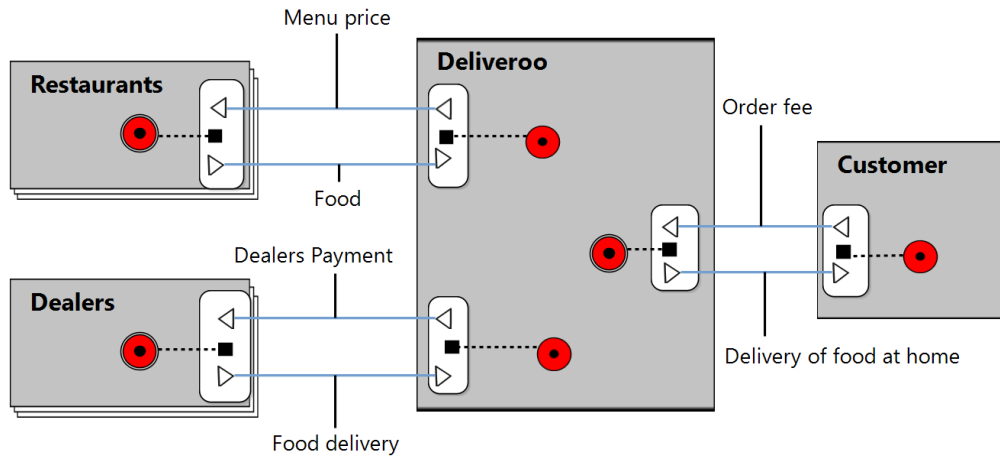
- Correspondence Analysis
  - Automatic generation of (partial) models
  - Relationships models



- 3 types
  - Business Models relationships
  - Business Process Models relationships
  - Business and Business Process Models relationships





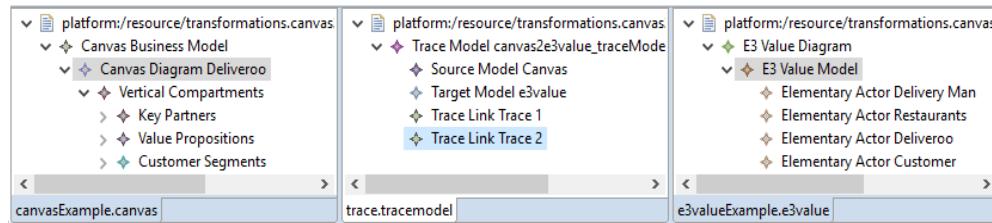
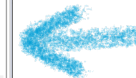
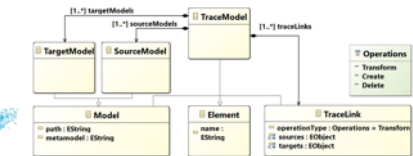
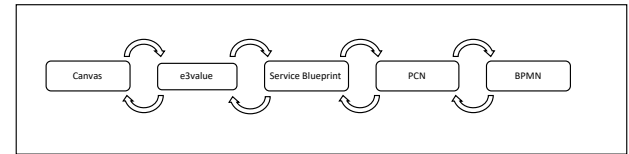






## Sirius

## epsilon



	Canvas from e <sup>3</sup> value	e <sup>3</sup> value from canvas	e <sup>3</sup> value from SBP	SBP from e <sup>3</sup> value	BPM from PCN	PCN from SBP	PCN from BPMN	BPMN from PCN
Case 1	32,00	68,97	3,45	8,70	100,00	44,64	85,71	60,24
Case 2	38,71	76,56	3,13	8,00	100,00	45,16	93,55	75,00
Case 3	47,83	80,77	3,85	9,52	95,24	45,65	91,30	73,17
Case 4	28,13	79,41	2,94	6,67	93,33	49,18	90,16	73,53
Case 5	37,14	72,22	2,78	5,88	100,00	40,28	87,50	55,36
AVG	36,76	75,59	3,23	7,75	97,71	44,98	89,65	67,46

**Universidad Rey Juan Carlos**

## Formal Support of Process Chain Networks using Model-driven Engineering and Petri nets

Elena Gómez-Martínez<sup>1</sup>, Francisco Pérez-Blanco<sup>2</sup>, Juan de Lara<sup>1</sup>, Juan Manuel Vara<sup>2</sup>, Esperanza Marcos<sup>2</sup>

<sup>1</sup>Universidad Autónoma de Madrid (mariaelena.gomez, juan.delara)@uam.es  
<sup>2</sup>Universidad Rey Juan Carlos (francisco.perez, juanmanuel.vara, esperanza.marcos)@urjc.es

**Context**

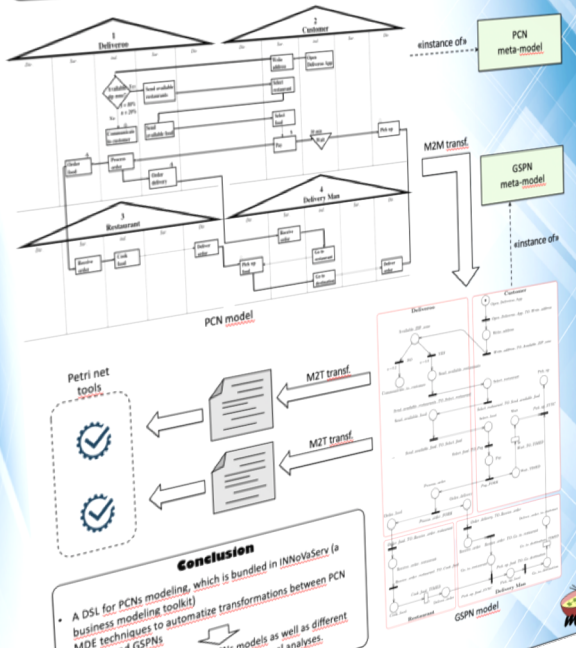
- Business Process Models as essential tools for competitive enterprises.
- Several notations proposed so far: BPMN, Service Blueprints, PCN, etc.

**Challenge**

- Some notations, like PCN, lacks tool support and formal semantics to enable modeling, verification and validation of process models.

**Approach**

- MDE techniques are used to map PCN models into Stochastic Petri Nets, for which several analysis tools exist.



**Conclusion**

- A DSL for PCNs modeling, which is bundled in INNoVaServ (a business modeling toolkit)
- MDE techniques to automate transformations between PCN models and GSPNs
- Enabling formal validation of PCNs models as well as different structural, functional and performance formal analyses.

SAC 2019  
Limassol, Cyprus  
April 8-12, 2019

**kybele**

**UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH**

**ICSOC 2017**

**Universidad Rey Juan Carlos**

ICSOC 2017  
International Conference on Service-Oriented Computing  
November 13-16, 2017. Málaga, Spain

## Validation of Service Blueprint models by means of formal simulation techniques

Montserrat Estañol<sup>1</sup>, Esperanza Marcos<sup>2</sup>, Xavier Oriol<sup>1</sup>, Francisco J. Pérez<sup>2</sup>, Ernest Teniente<sup>1</sup>, Juan M. Vara<sup>2</sup>

<sup>1</sup>Information Modeling & Processing  
Universitat Politècnica de Catalunya  
Barcelona – Spain  
(estanyol, oriol, teniente)@essi.upc.edu  
@ErnestTeniente

<sup>2</sup>Kybele Research Group  
Universidad Rey Juan Carlos  
Madrid – Spain  
(esperanza.marcos, francisco.perez, juanmanuel.vara)@urjc.es  
@EMarcosMtnz | @jmvara

**kybele**

① Kybele at a glance

② INNoVaServ

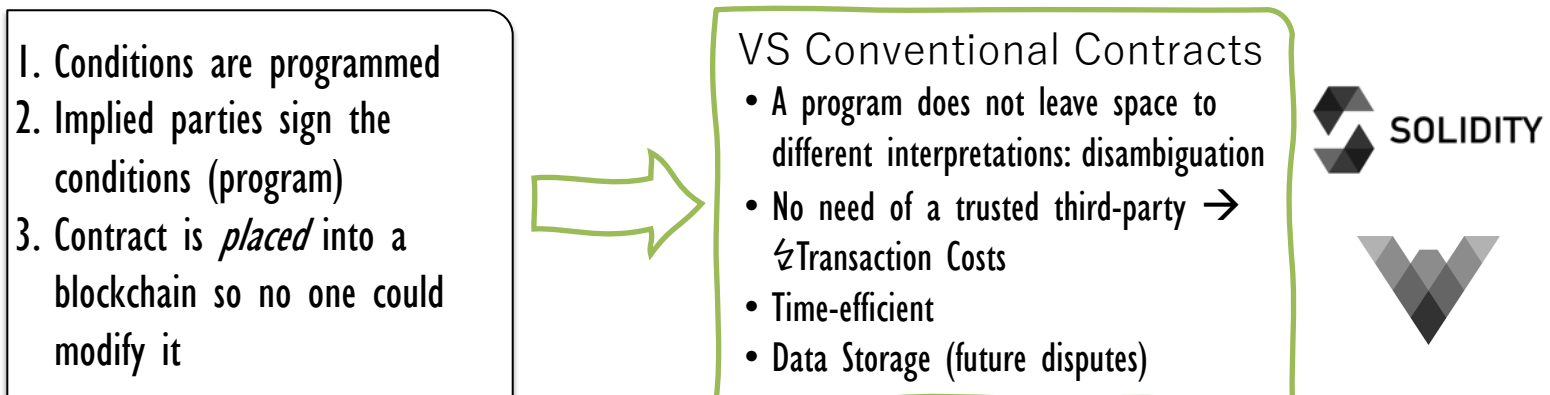
③ SmaC

- What is a Blockchain?
  - A distributed DB + Encryption + Immutability + stored procedures (smart contracts)
    - A blockchain is a list (chain) of groups (blocks) of transactions
    - Like traditional DDBBs they can be used for anything a DB is used.
- How does it work?
  - Interested subjects add transactions to the pool
  - Nodes verify and add them to some block on the ledger
  - Ledger is replicated among distributed nodes
  - Eventual consistency
    - In the absence of centralized control, all nodes eventually achieve consensus about the content of the ledger
  - Append-only data structure
    - May add transactions – Nearly imposible to change data



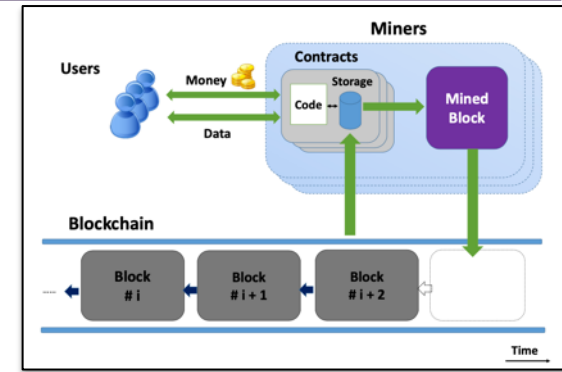
- Computer programs
  - Hosted on the BC (Ethereum)
  - Executes **autonomously** the clauses collected in it when the conditions are satisfied
    - DTL as a DDBB
    - Smart Contracts as triggers or microservices where the **business logic** transacting with that data lives
  - Blockchain technology “Sets in stone” the agreement
    - The contract inherits trust-less, immutability, transparency ...

Szabo, N. (1996). Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought*, (16), 18, 2.



An instance of program code that runs in the blockchain

Program code | Storage file | Account balance



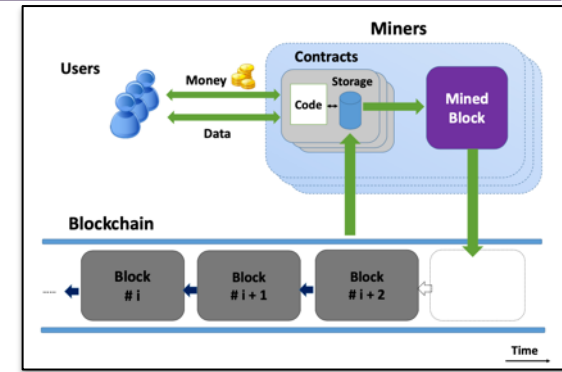
(Delmolino et al., 2016)

## SMART CONTRACT MODEL

1. User create the contract: transaction posting
  - a) Code cannot be changed
  - b) Storage file stored in the blockchain
2. Contract is executed upon message received (either users or contracts)
  - a. Read/Write from its file
  - b. Receive/Send money from its account balance from/to users (contracts)
3. Miners reach consensus on the output of the execution and update the blockchain accordingly

An instance of program code that runs in the blockchain

Program code | Storage file | Account balance



(Delmolino et al., 2016)

## CONTRACT INVOCATION – $T_x$ AS FUNCTION CALLS

- Contract code will be invoked whenever it receives a  $T_x$  from a user
- Multiple entry points of execution – each one is defined as a function
  - After processing the message, contract can return value back to the sender
- The content of the  $T_x$  will specify the entry point at which the contract's will be invoked

```

1 pragma solidity >=0.4.22 <0.6.0;
2
3 contract Purchase{
4     uint public val;
5     address payable public seller;
6     address payable public buyer;
7
8     enum State{Created,Locked,Inactive};
9     State public state;
10
11     constructor()public{
12         seller = msg.sender;
13         val = msg.value /2;
14         require((2 * val) == msg.value,"Value has to be even.");
15     }
16
17     modifier condition(bool condition){
18         require(condition);
19         _;
20     }
21
22     modifier onlyBuyer(){
23         require(msg.sender == buyer,"Only buyer can call this.");
24         _;
25     }
26
27     modifier onlySeller(){
28         require(msg.sender == seller,"Only seller can call this.");
29         _;
30     }
31
32     modifier inState(State _state){
33         require(state == _state);
34         _;
35     }
36
37     event Aborted();
38     event PurchaseConfirmed();
39     event ItemReceived();
40
41     function abort()public onlySeller inState(State.Created){
42         emit Aborted();
43         state = State.Inactive;
44         seller.transfer(address(this).balance);
45     }
46
47     function confirmPurchase() public inState(State.Created) condition(msg.value == (2 * val)) {
48         emit PurchaseConfirmed();
49         buyer = msg.sender;
50         state = State.Locked;
51     }
52
53     function confirmReceived() public onlyBuyer inState(State.Locked){
54         emit ItemReceived();
55         state = State.Inactive;
56         buyer.transfer(val);
57         seller.transfer(address(this).balance);
58     }
59

```

Compiler version

Name

State variables

Constructor

Modifiers

Events

Functions

Libraries & Interfaces
Global variables
Events / Modifiers
Contract signature ( <i>is</i> )
Constructor
Functions

Similar to a Class in  
any OOPL



## Learning Curve

- Alharby, M., Aldweesh, A., & van Moorsel, A. (2018). Blockchain-based smart contracts: A systematic mapping study of academic research (2018). In *Proceedings of the 2018 International Conference on Cloud Computing, Big Data and Blockchain*.

## IT – Business Gap

- Mik, E. (2017). Smart contracts: terminology, technical limitations and real world complexity. *Law, Innovation and Technology*, 9(2), 269-300.
- Bosu, A., Iqbal, A., Shahriyar, R., & Chakraborty, P. (2019). Understanding the motivations, challenges and needs of blockchain software developers: A survey. *Empirical Software Engineering*, 24(4), 2636-2673.

## Security Issues

- Mavridou, A., & Laszka, A. (2018, February). Designing secure ethereum smart contracts: A finite state machine based approach. In *International Conference on Financial Cryptography and Data Security* (pp. 523-540). Springer, Berlin, Heidelberg.

“In other words, they're code that does what it's been programmed to do.

If the **business rules** ... have been defined badly and/or the programmer doesn't do a good job, the result is going to be a mess, and, even if programmed correctly, a smart contract isn't smart – it just functions as **designed**.”

What's a smart contract (and how does it work)?  
*Computer World*, Jul 29 (2019)

- [Formal] verification of Smart Contracts

Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., ... & Zanella-Béguelin, S. (2016, October). Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security* (pp. 91-96). ACM.

Bragagnolo, S., Rocha, H., Denker, M., & Ducasse, S. (2018, March). SmartInspect: solidity smart contract inspector. In *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 9-18). IEEE.

- DSL-based
  - Legal principles-based DSL (Adico-Solidity).
  - Natural language-based (SmaCoNat)

Frantz, C. K., & Nowostawski, M. (2016, September). From institutions to code: Towards automated generation of smart contracts. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self\* Systems (FAS\* W)* (pp. 210-215). IEEE.

Regnath, E., & Steinhorst, S. (2018, September). SmaCoNat: Smart Contracts in Natural Language. In *2018 Forum on Specification & Design Languages (FDL)* (pp. 5-16). IEEE.

- Templates for Smart Contracts

Clack, C. D., Bakshi, V. A., & Braine, L. (2016). Smart contract templates: foundations, design landscape and research directions. *arXiv preprint arXiv:1608.00771*.

- MDE-based
  - Both use MDE to map the business process (BPMN) into a smart contract.
  - Lorikeet need to extend the BPMN notation (2 elements)

López-Pintado, O., García-Bañuelos, L., Dumas, M., & Weber, I. (2017, September). Caterpillar: A Blockchain-Based Business Process Management System. In *BPM (Demos)*.

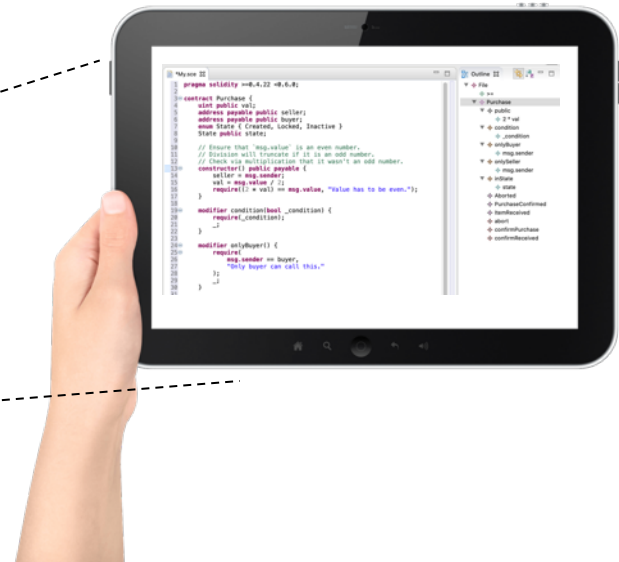
Tran, A. B., Lu, Q., & Weber, I. (2018). Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management. In *BPM (Dissertation/Demos/Industry)* (pp. 56-60).

Raise the level of abstraction at which  
Smart Contracts are developed /designed

MDE TO THE RESCUE



**SmaC**

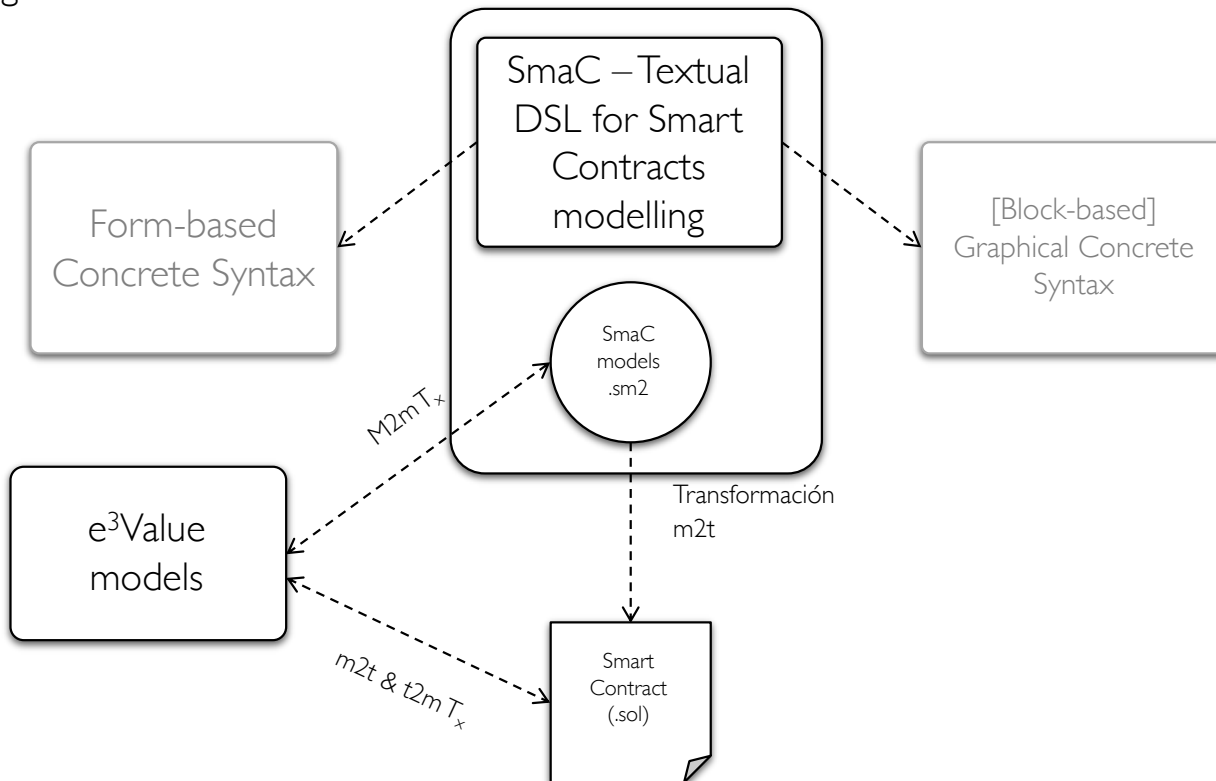


Methodological  
Proposal

Smart Contracts  
specification  
process

Custom Clauses  
specification process

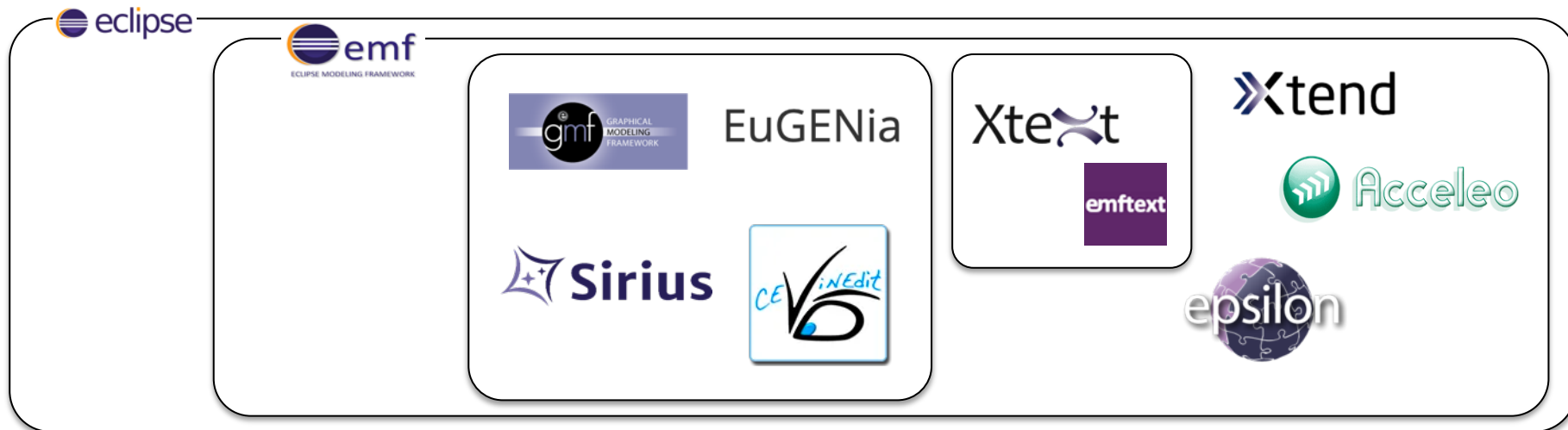
Technological  
Proposal



- What is Xtext?

- Framework for textual DSLs development
- Xtend (Java-like) for the development of validations, quickfixes, etc.
- Ecore metamodel automatically generated from the grammar.

Xtext



- How to develop a textual language?

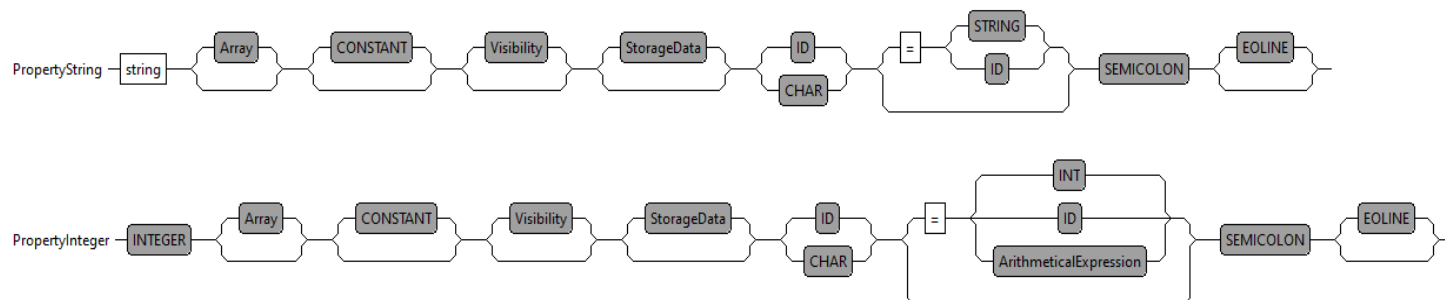
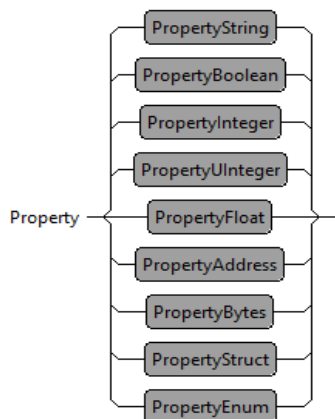
- Write the grammar

a) Define the terminals.

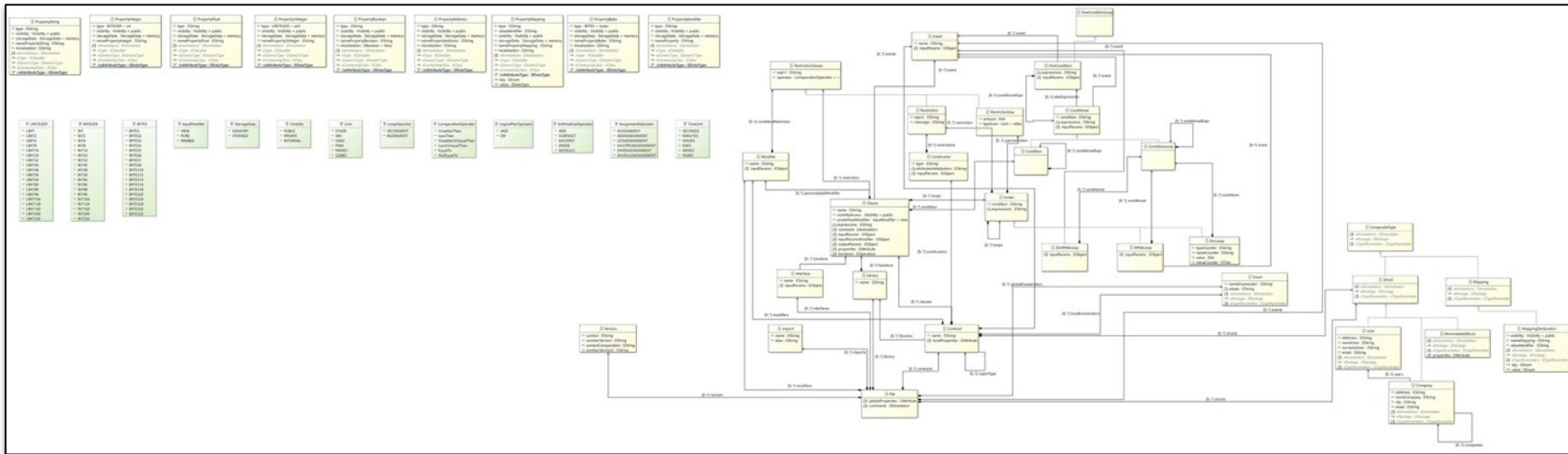
```
terminal SEMICOLON returns ecore::EChar:
';
;
terminal DOT returns ecore::EChar:
'.'
;
;
```

b) Define the rules.

```
Property returns ecore::EAttribute:
PropertyString|PropertyBoolean|PropertyInteger|PropertyUInteger|PropertyFloat|PropertyAddress|PropertyBytes|PropertyStruct|PropertyEnum
;
PropertyString:
type= "string" Array? CONSTANT? visibility = Visibility? (storageData = StorageData)? (namePropertyString = ID|CHAR) ('=' initialization = (STRING|ID))? SEMICOLON EOLINE?
;
PropertyInteger:
type = INTEGER Array? CONSTANT? visibility = Visibility? (storageData = StorageData)? (namePropertyInteger = ID|CHAR) ('=' (INT|ID|ArithmeticalExpression))? SEMICOLON EOLINE?
;
```

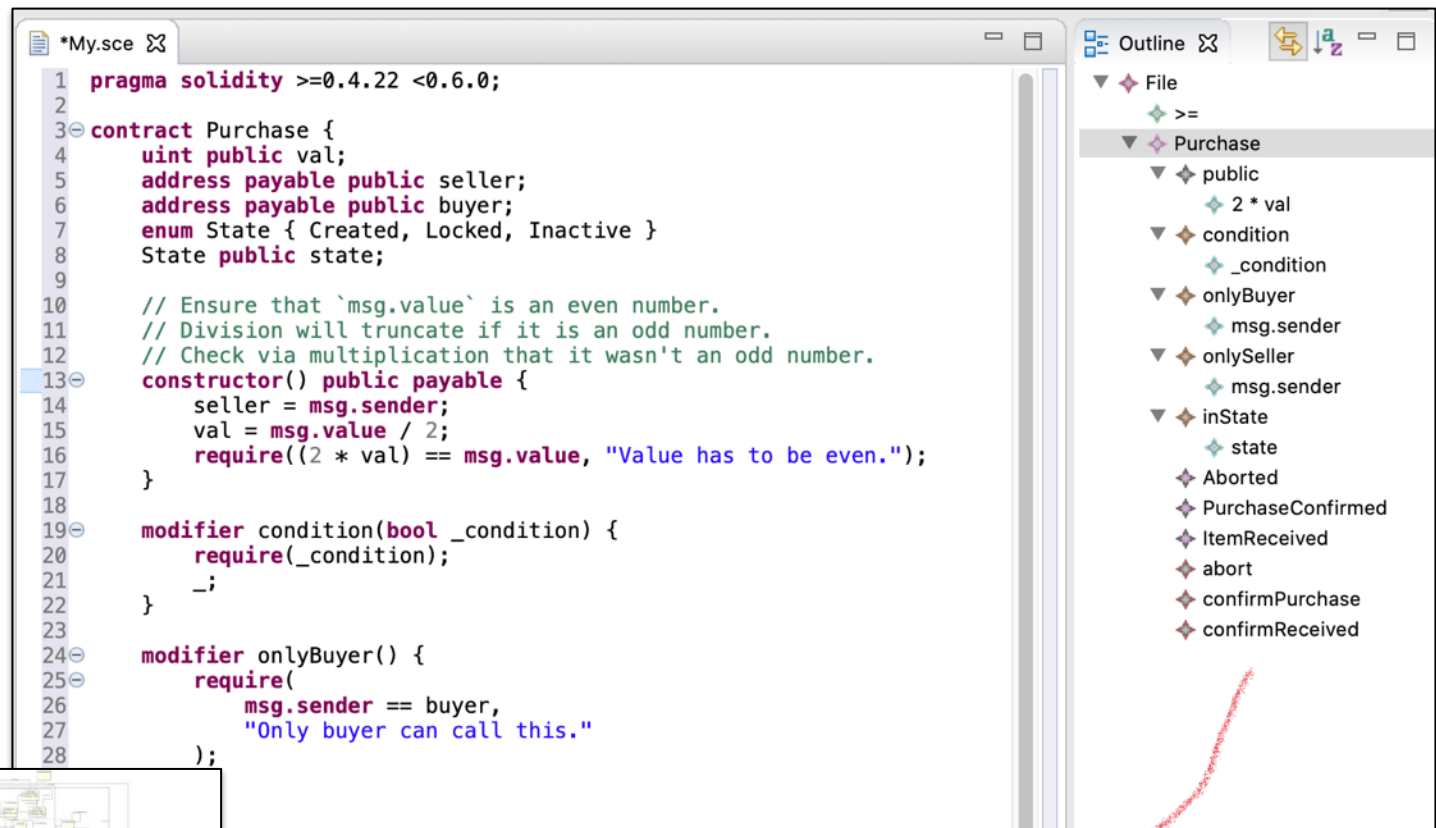


3. Generate language artifacts.





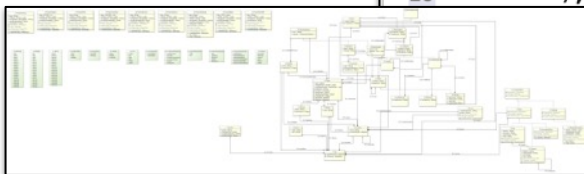
- How to develop a textual language?
  - Run the Generated Eclipse plug-in.



```

1 pragma solidity >=0.4.22 <0.6.0;
2
3 contract Purchase {
4     uint public val;
5     address payable public seller;
6     address payable public buyer;
7     enum State { Created, Locked, Inactive }
8     State public state;
9
10    // Ensure that `msg.value` is an even number.
11    // Division will truncate if it is an odd number.
12    // Check via multiplication that it wasn't an odd number.
13    constructor() public payable {
14        seller = msg.sender;
15        val = msg.value / 2;
16        require((2 * val) == msg.value, "Value has to be even.");
17    }
18
19    modifier condition(bool _condition) {
20        require(_condition);
21    }
22
23    modifier onlyBuyer() {
24        require(
25            msg.sender == buyer,
26            "Only buyer can call this."
27        );
28    }

```



Conforms To

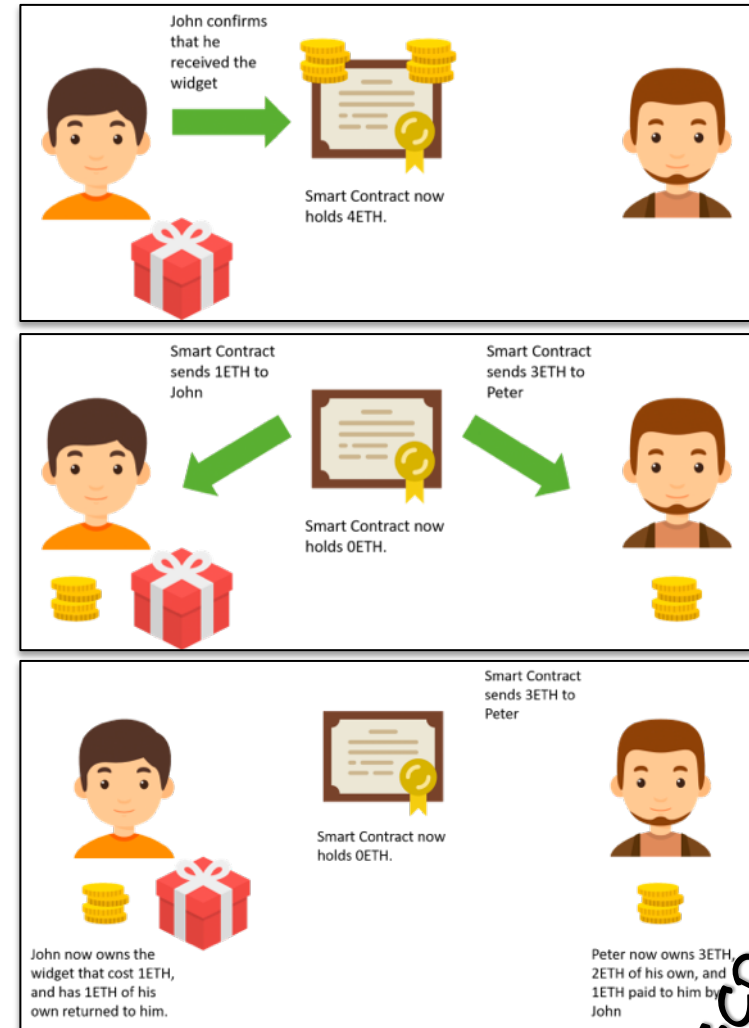
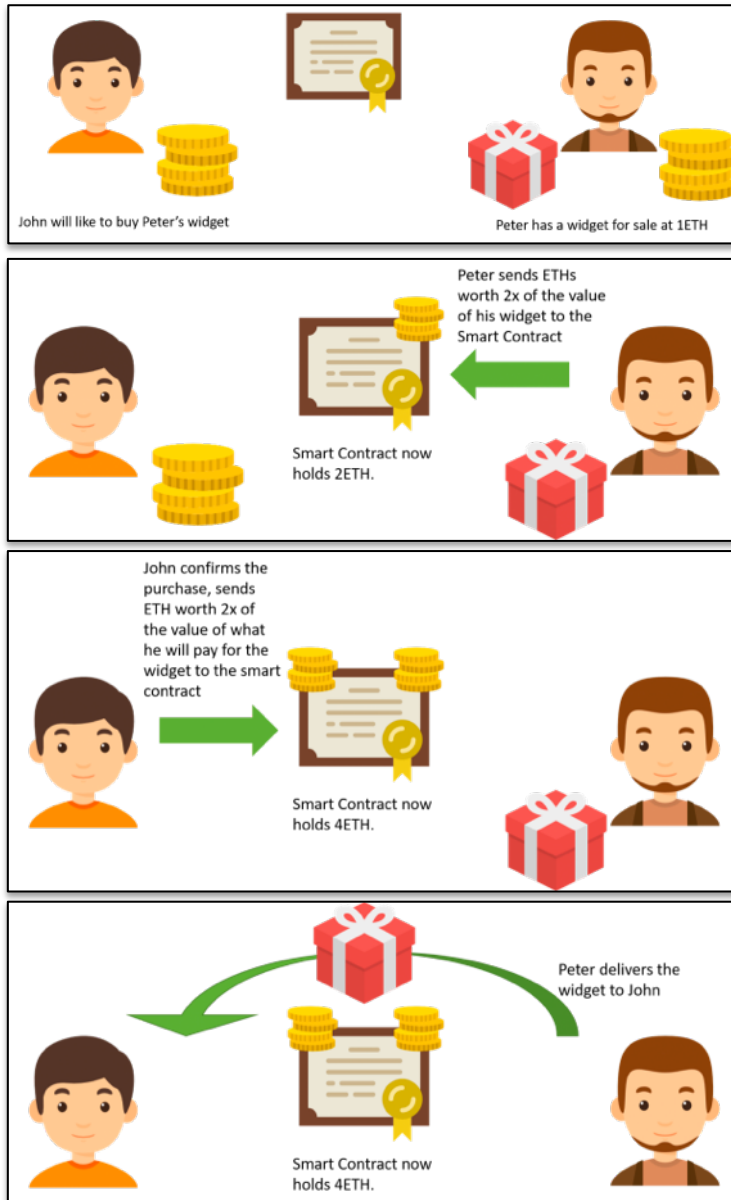
- How to develop a textual language?
  5. [Generate Code Generator - Xtend]
  6. [Unit Testing]
  7. [Creating Custom Validation Rules]

```
@Check
def checkContractStartsWithCapital(Contract imports) {
    if (!Character.isUpperCase(imports.name.charAt(0))) {
        error('Contract's name should start with a capital',
            SM2Package.Literals.CONTRACT__NAME,
            INVALID_NAME)
    }
}
```

Including  
quickfixes

```
@Fix(SM2Validator.INVALID_NAME)
def capitalizeName(Issue issue, IssueResolutionAcceptor acceptor) {
    acceptor.accept(issue, 'Capitalize name', 'Capitalize the name.', 'upcase.png') [
        context |
        val xtextDocument = context.xtextDocument
        val firstLetter = xtextDocument.get(issue.offset, 1)
        xtextDocument.replace(issue.offset, 1, firstLetter.toUpperCase)
    ]
}
```

# Example: Safe Remote Purchase



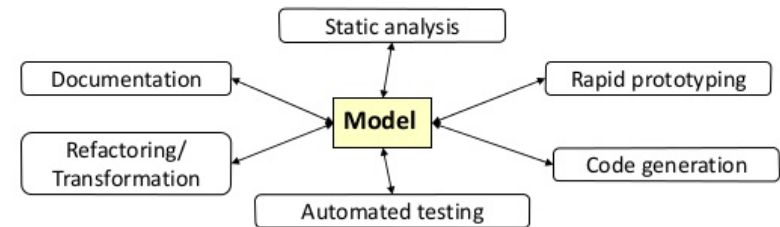
- [SmaC in action](#)
- [Safe Remote Purchase](#) in Solidity documentation

**ESCROW SERVICES**

- Define and enrich customized textual structures
  - E.g: gas control in loops to avoid

- Reduce the learning curve

- Auto-completion
- Syntactical validation
- QuickFixes
- Good practices
- Auto-documentation ...



[Illustration by Bernhard Rumpe]

Cabot, J. Lightweight Model-Driven Engineering. Les journées nationales du GDR GPL. Jun 15, 2017

- Development of technological bridges
  - Close the gap between business professionals and developers



Business and Business Process Modeling toolkit for Service Design

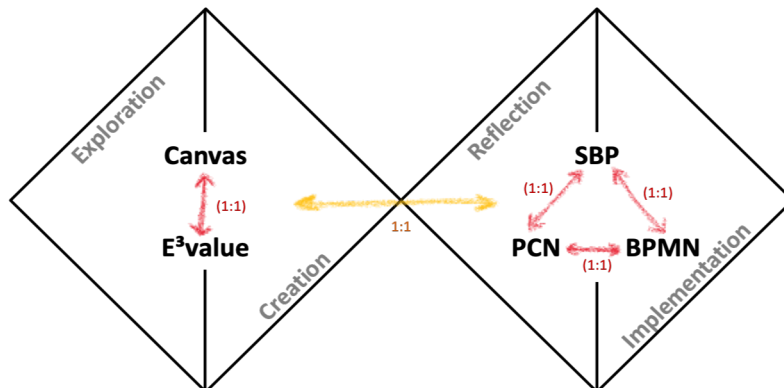
**Canvas**

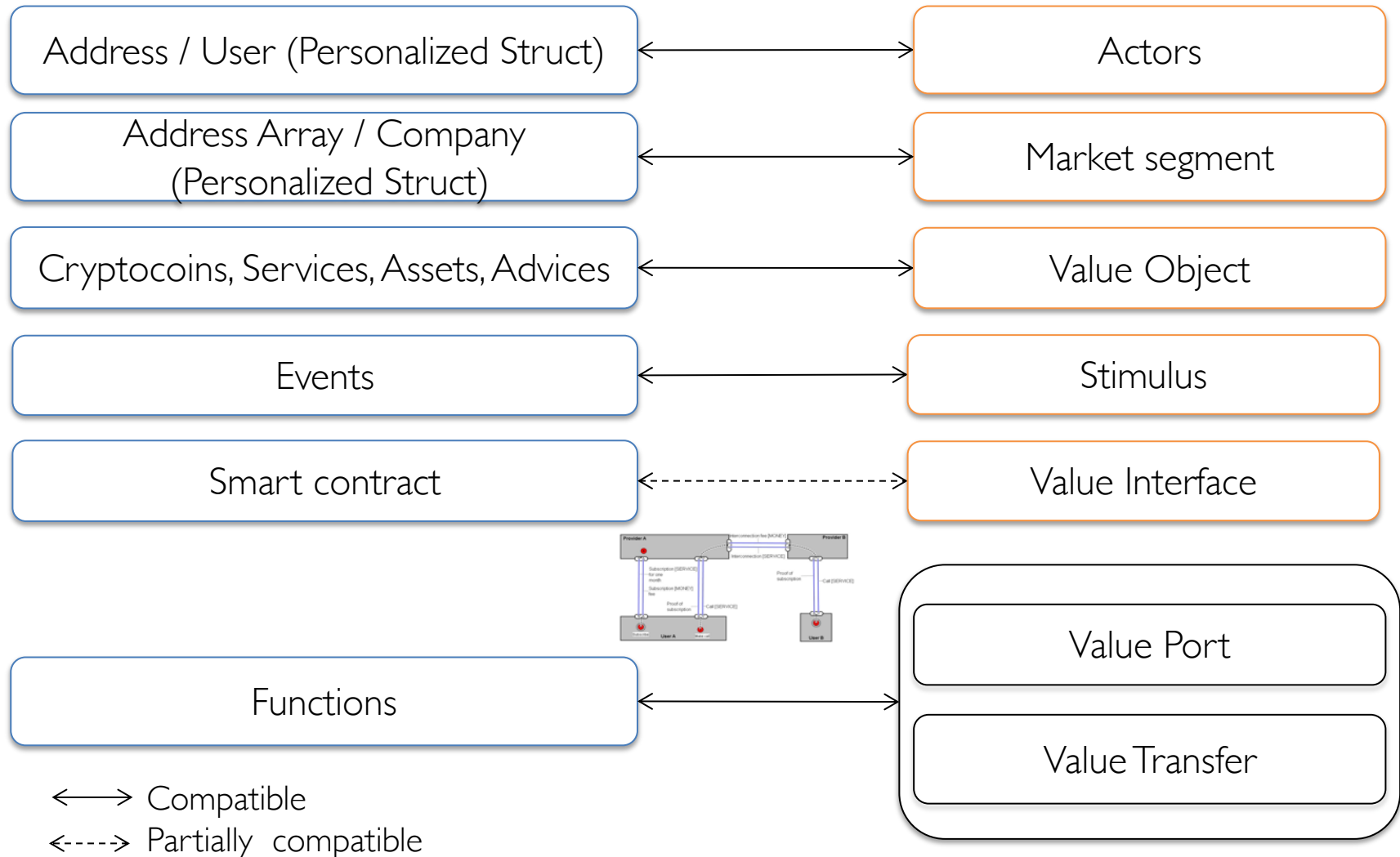
**e<sup>3</sup>value**

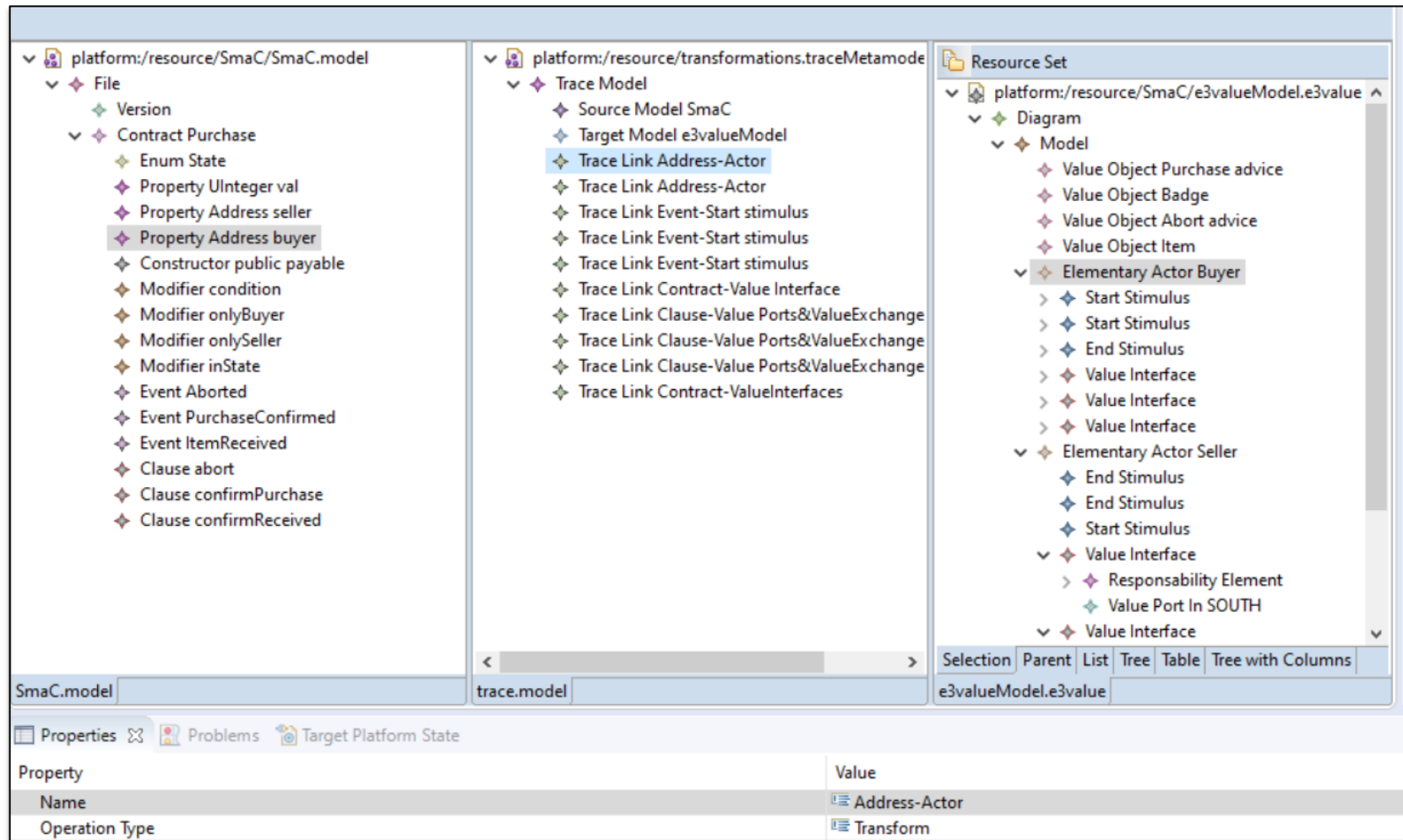
**PCN**

**Service  
Blueprint**

**BPMN**







The screenshot displays a modeling tool interface with three main panels and a bottom table.

**Left Panel (SmaC.model):**

- platform:/resource/SmaC/SmaC.model
  - File
    - Version
  - Contract Purchase
    - Enum State
    - Property UInteger val
    - Property Address seller
    - Property Address buyer
    - Constructor public payable
    - Modifier condition
    - Modifier onlyBuyer
    - Modifier onlySeller
    - Modifier inState
    - Event Aborted
    - Event PurchaseConfirmed
    - Event ItemReceived
    - Clause abort
    - Clause confirmPurchase
    - Clause confirmReceived

**Middle Panel (traceMetamodel):**

- platform:/resource/transformations.traceMetamodel
  - Trace Model
    - Source Model SmaC
    - Target Model e3valueModel
    - Trace Link Address-Actor
    - Trace Link Address-Actor
    - Trace Link Event-Start stimulus
    - Trace Link Event-Start stimulus
    - Trace Link Event-Start stimulus
    - Trace Link Contract-Value Interface
    - Trace Link Clause-Value Ports&ValueExchange
    - Trace Link Clause-Value Ports&ValueExchange
    - Trace Link Clause-Value Ports&ValueExchange
    - Trace Link Contract-ValueInterfaces

**Right Panel (Resource Set):**

- Resource Set
  - platform:/resource/SmaC/e3valueModel.e3value
    - Diagram
      - Model
        - Value Object Purchase advice
        - Value Object Badge
        - Value Object Abort advice
        - Value Object Item
        - Elementary Actor Buyer
          - Start Stimulus
          - Start Stimulus
          - End Stimulus
          - Value Interface
          - Value Interface
          - Value Interface
        - Elementary Actor Seller
          - End Stimulus
          - End Stimulus
          - Start Stimulus
          - Value Interface
            - Responsability Element
              - Value Port In SOUTH
          - Value Interface

**Bottom Table:**

Property	Value
Name	Address-Actor
Operation Type	Transform

- SmaC validation
- Technological Bridges development
  - m2m & m2t transfos.
- Graphical concrete syntaxes development
- Extend e<sup>3</sup>Value with smart contract elements non directly matched
  - Modifiers.
- V&V of
  - High-level specifications
  - Transfos.
- Enable automatic deploying mechanisms

