Developing an Ontology of Software Evolution

Preliminary research results

Xiaowei Wang

ICT International Doctoral School,
University of Trento, Italy
xwang@disi.unitn.it

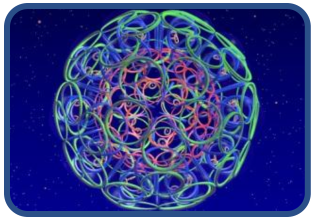
Outline

1. Motivation
2. Basic intuitions
3. Concept of "Software"
4. Concept of "Species"
5. Evolution, Maintenance and Adaptation
6. Conclusion
7. Future work

Motivation

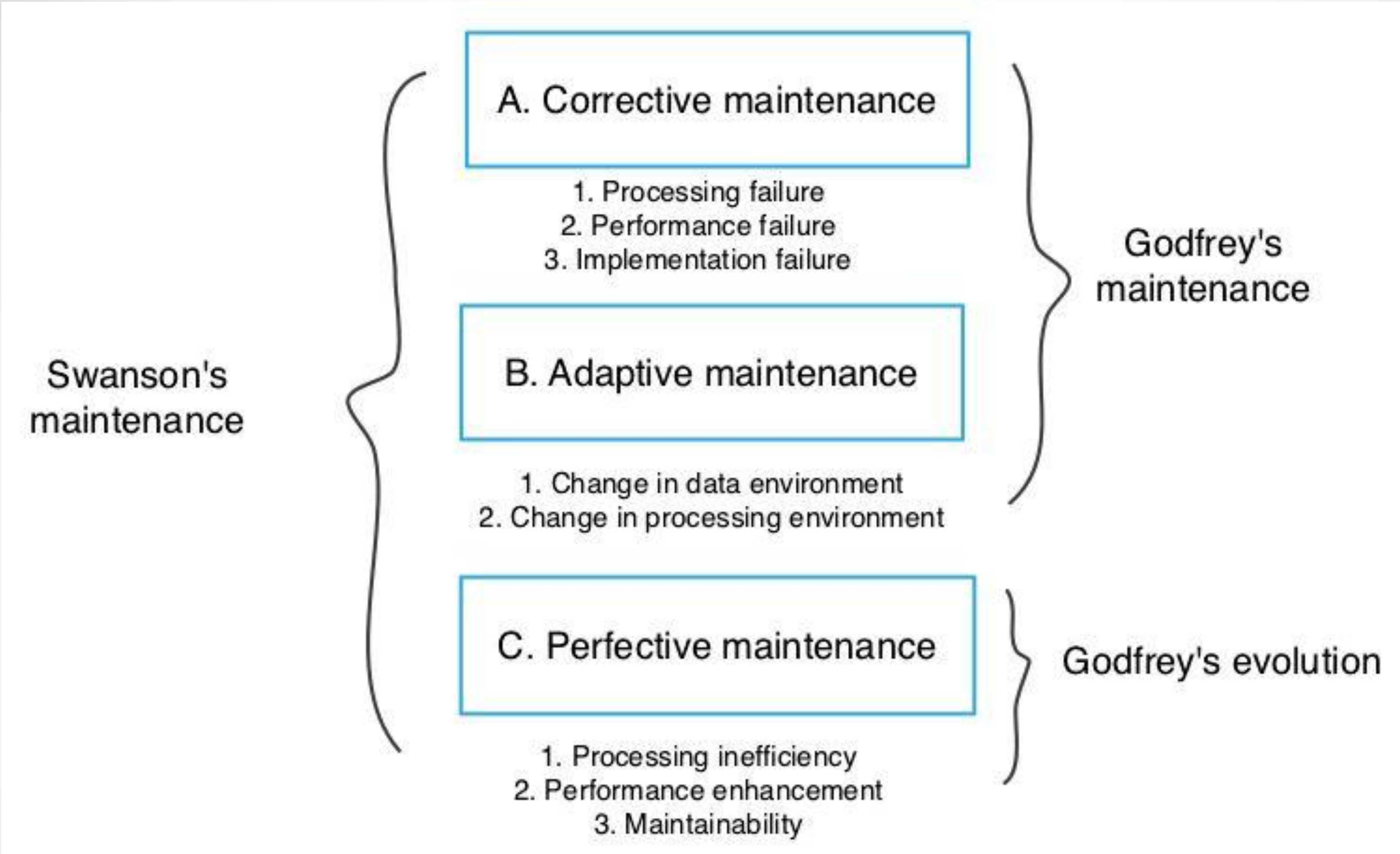
People rely on software heavily

Software changes rapidly



No universally shared concepts for software change

Different kinds of software change



As Godfrey states: Maintenance suggests preservation and fixing, whereas evolution suggests new designs evolving from old ones

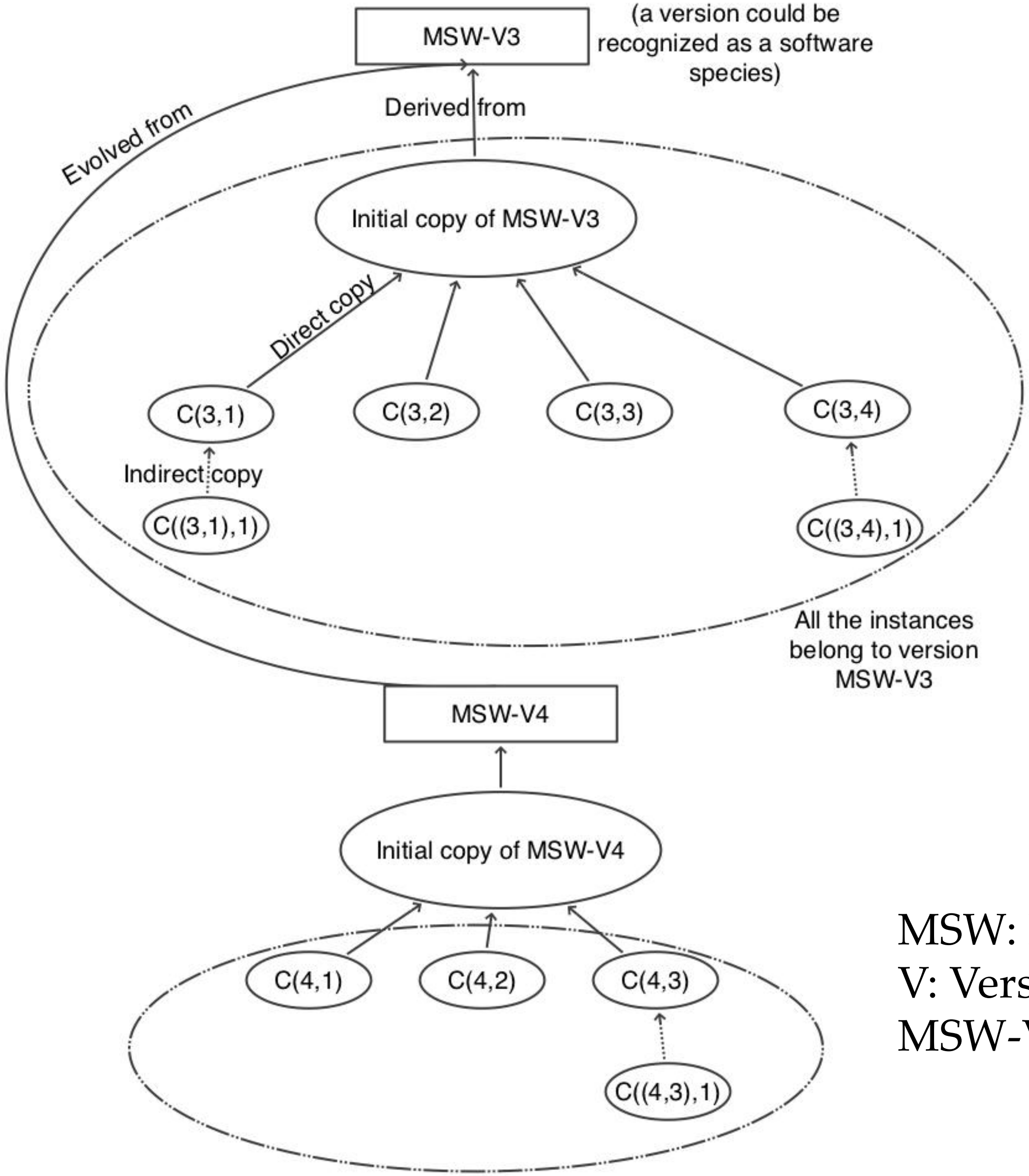
Basic intuitions

- Evolution only happens at species level

- Software Specifications = Software Species (laws)
- Software Species = Software Version (generally)
- Software (copy) = individual

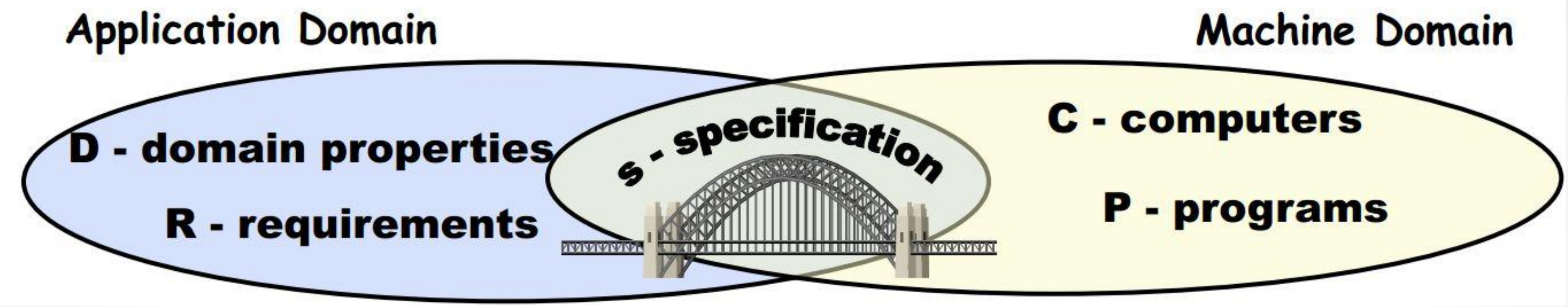
- Changes in software species are counted as software evolution

A example for intuition



MSW: Microsoft Word;
V: Version;
MSW-V3: Microsoft Word Verstion 3

A formula according to requirement engineering

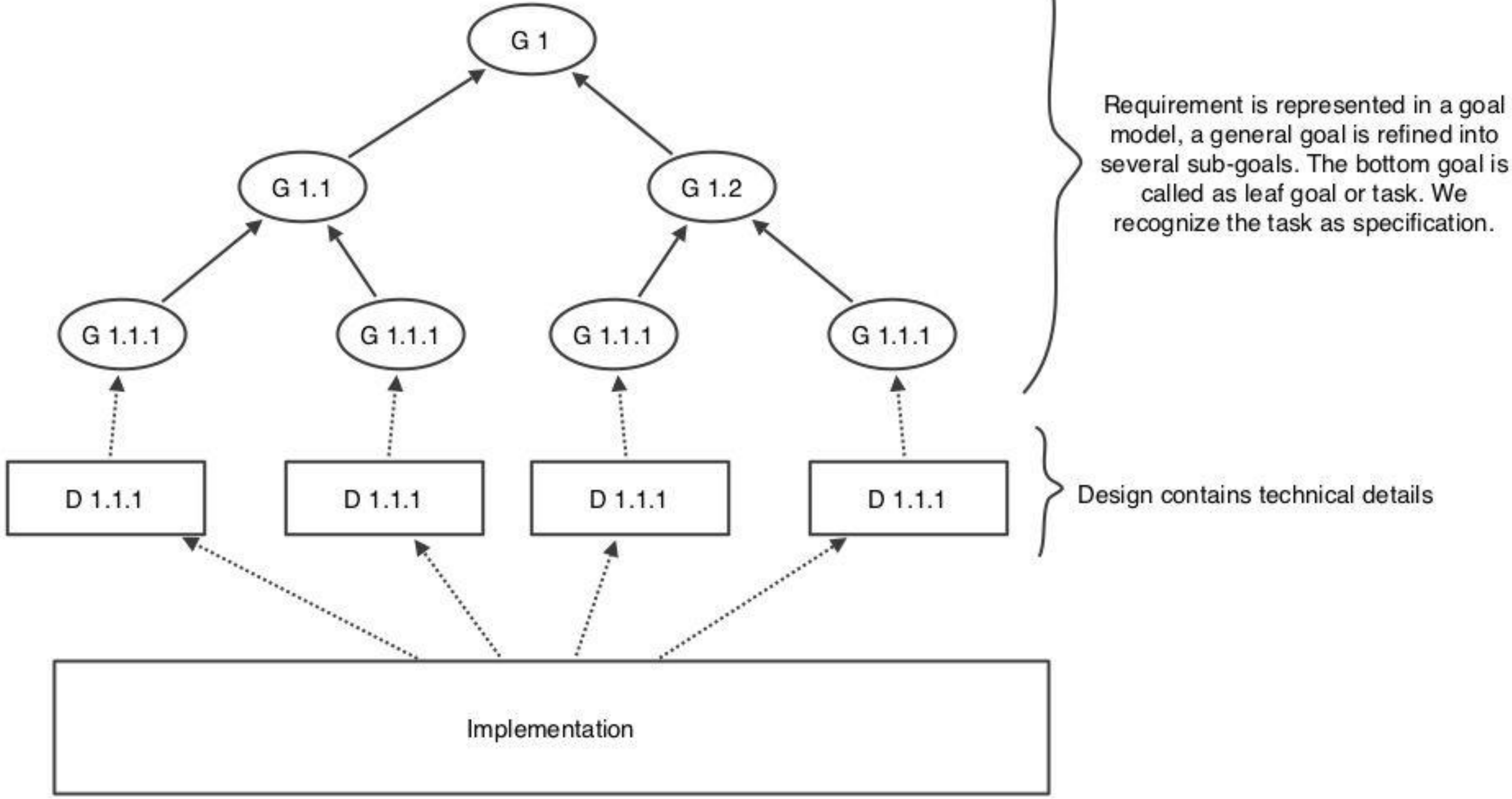


$$(D, S \vdash R) \wedge (D_{Des}, Des \vdash S) \wedge (D_I, I \vdash Des)$$

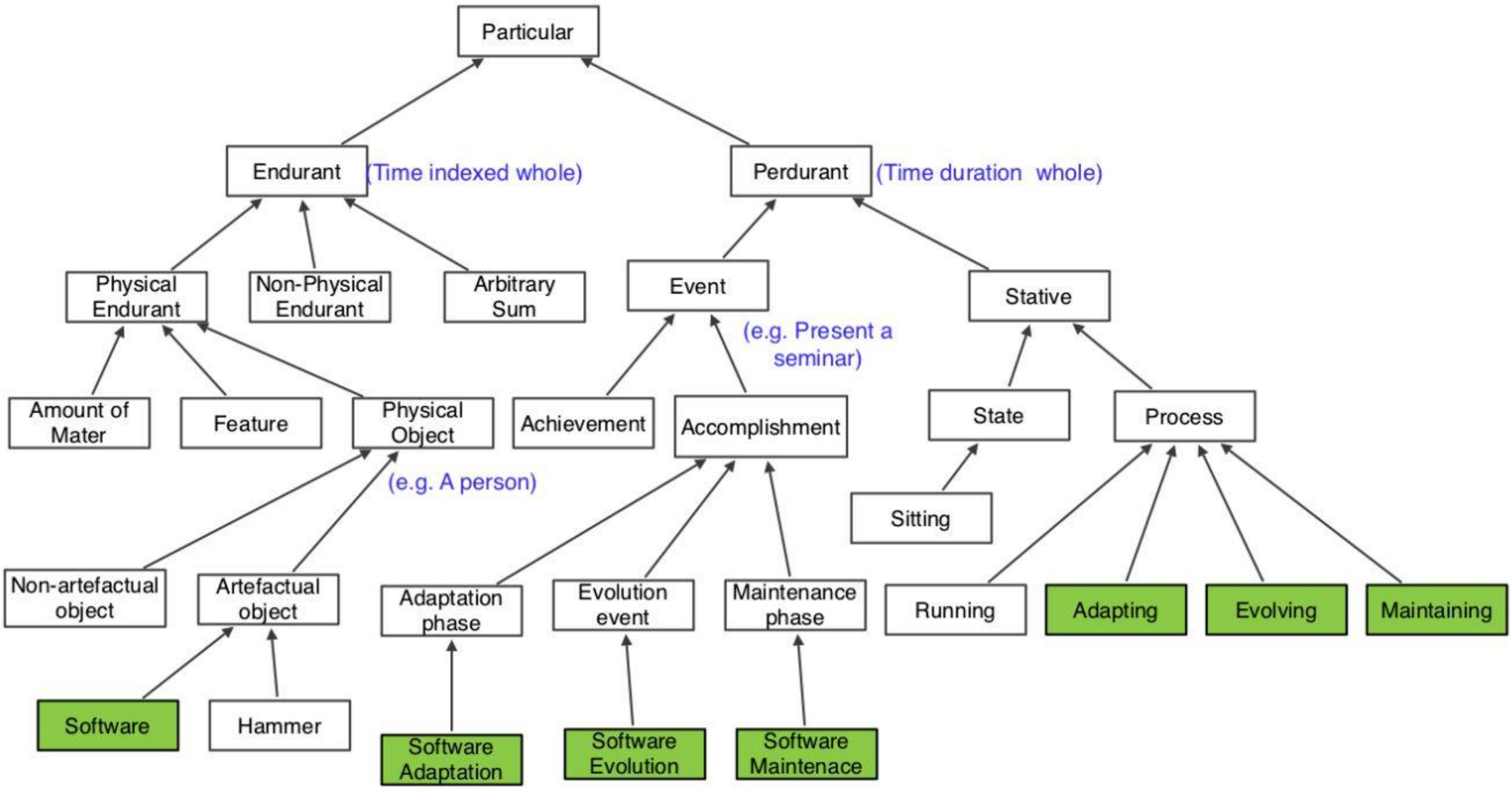
Abbreviation	Related concepts
D	Domain knowledge
R	Requirement
S	Specification
Des	Design
I	Implementation

A graphical explanation of the formula

$$(D, S \vdash R) \wedge (D_{Des}, Des \vdash S) \wedge (D_I, I \vdash Des)$$



A preliminary ontology of software evolution according to DOLCE



Concept of Software

- position:

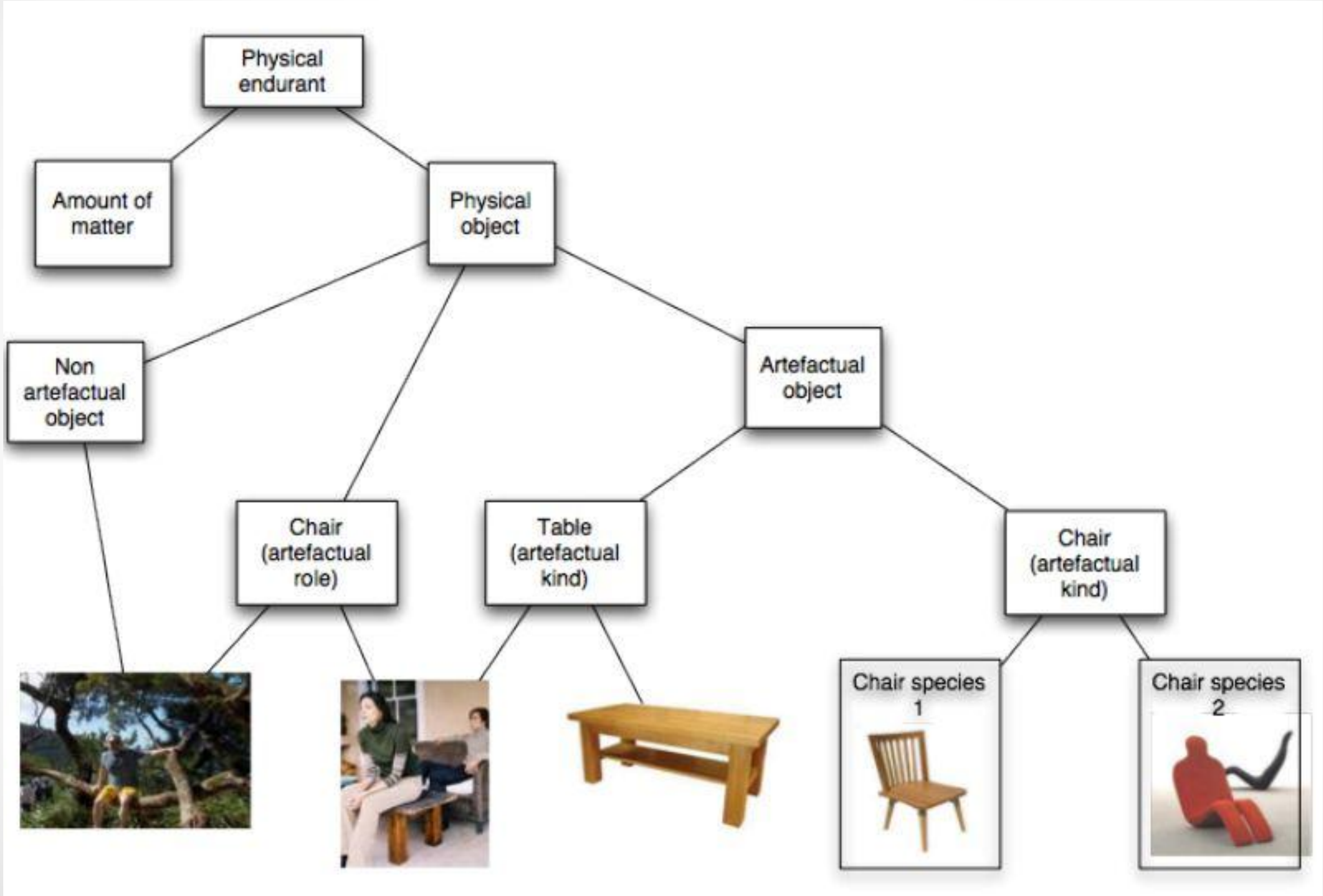
DOLCE:Physical Object(source code in harddisk)

- research target:

Software as DOLCE: Artefactual object (source code according to a design)

$$\exists x(Software(x) \rightarrow Artefactural\ object(x))$$

An ontology of Artefactual object



An comparison between Oberle's ontology and ours

Concetps according to Oberle's ontology

- Software ("SoftwareAsCode"):

  Encoding of an algorithm specification

  (e.g. C, Java, Python, pseudo code or in mind)

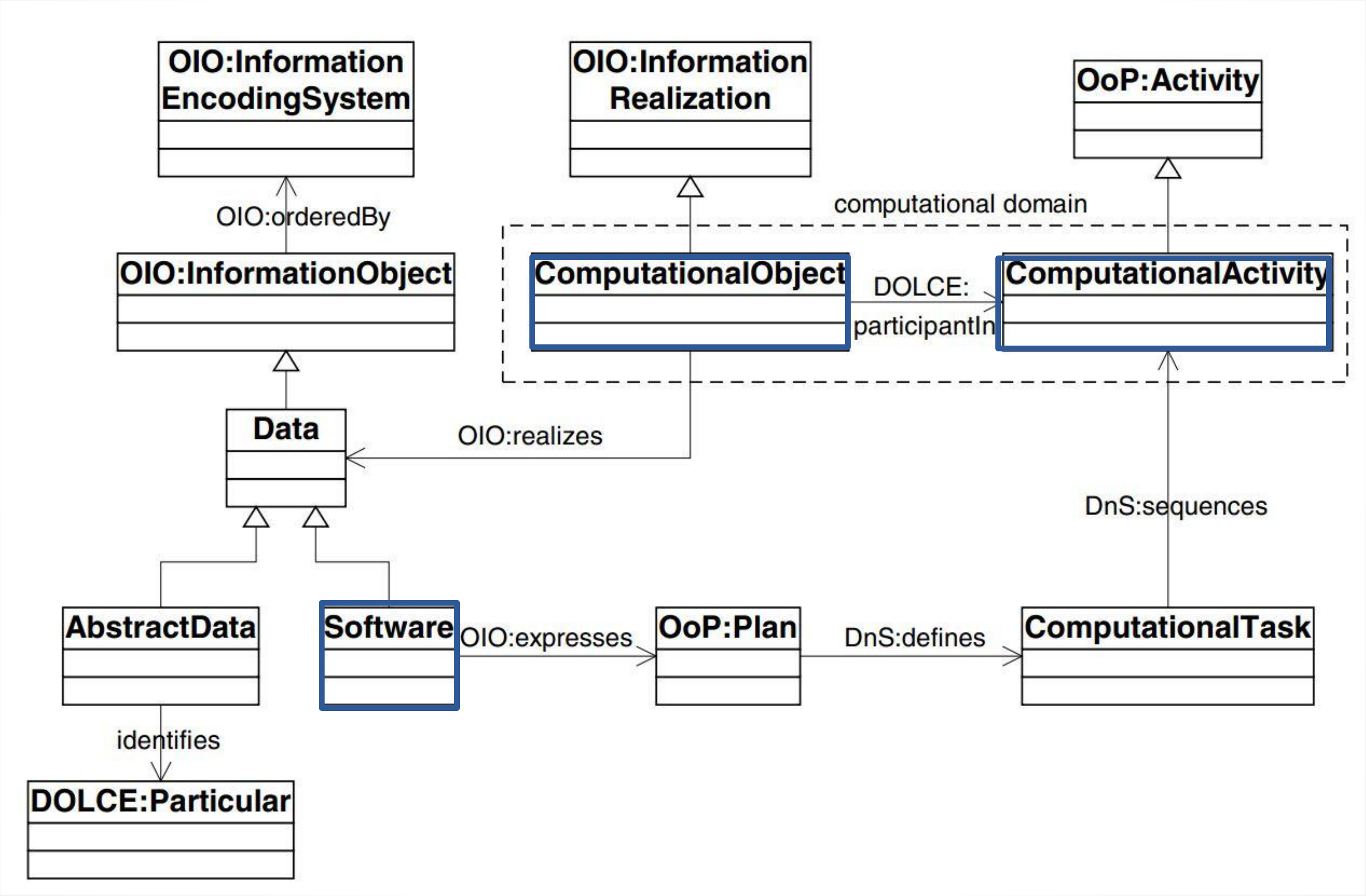
- ComputationalObjects:

  Realization of the code in a concrete hardware, and he positioned it in DOLCE framework as PhysicalEndurants

- ComputationalActivity

  The activities presented by the running system

An comparison between Oberle's ontology and ours



An comparison between Oberle's ontology and ours

Concepts from us	Concepts from Oberle	Comparison
Specification	No species level	
Design	SoftwareAsCode (encoding of algorithm)	"SoftwareAsCode" (despite in fact) actually more similar with "Design", it could be pseudo code or even algorithm in mind.
Software (copy) developed from Implementation	ComputationalObject (physicial existence on hard disk or memory card)	We prefer to call the realization of a design as a piece of software. It seems unintuitive we can not call a copy of Microsoft Word, for example, as a piece of software which is stored in a hard disk.
	ComputationalActivity (performance in running time)	We believe that "ComputationalActivity" is a suitable choice of this concept to represent the activities of software in running time, and we prefer to reuse this concept in our ontology.

Conept of Species

- A species is described as a "natural kind" according to Manhner's theory

- **Property** (something we can perceive or measure) (e.g. shapes, colors, sizes, weights, length …)
- **Laws** (something constraining the related properties) (e.g. thermometer)
- **Natural kind** (a set of shared laws)

if we focus on constantly related properties, we are able to find things possessing the same laws

Properties and laws

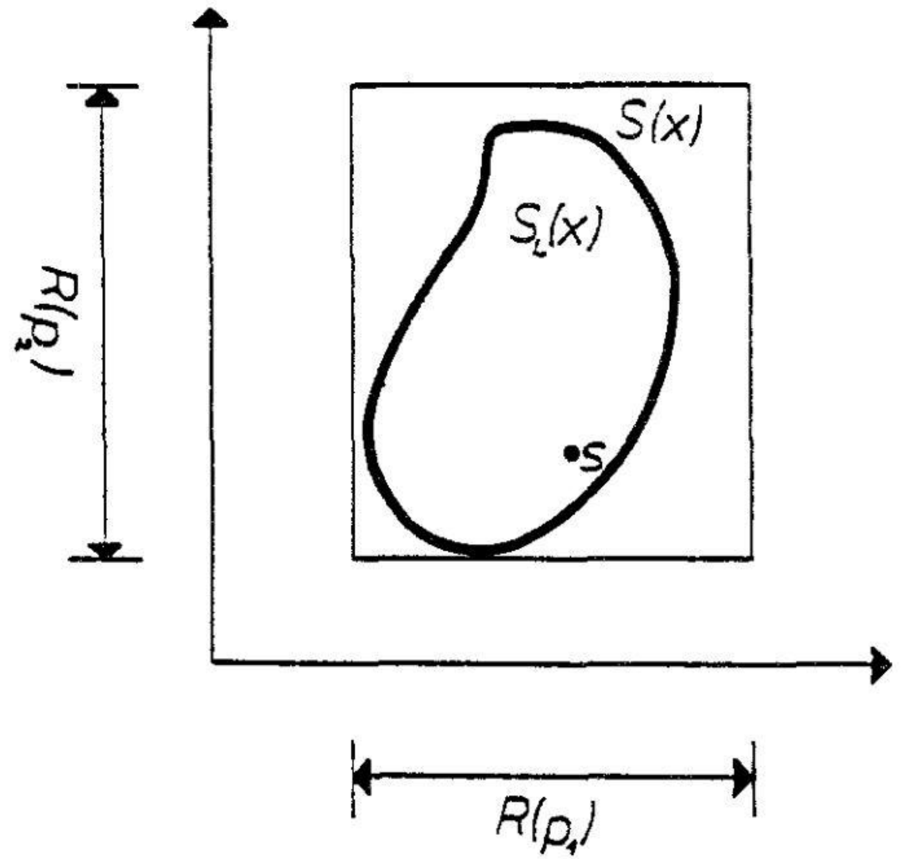
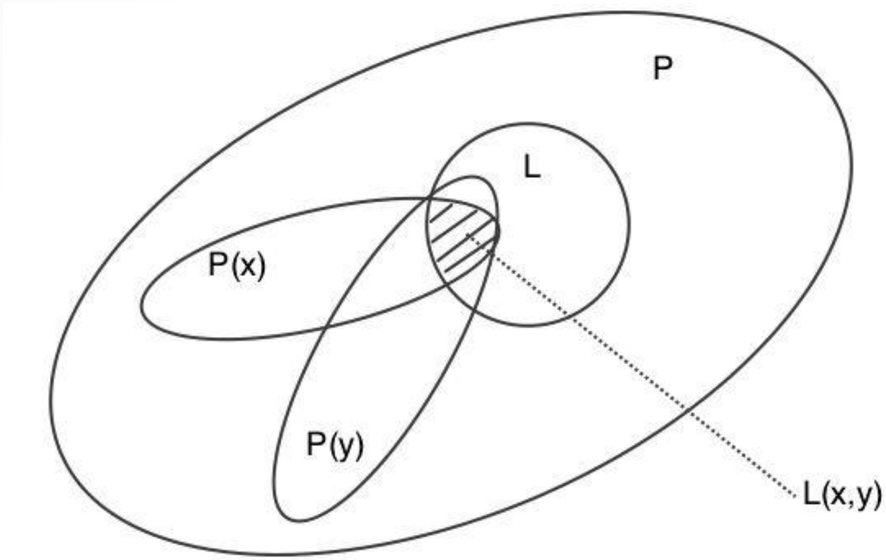


Fig. 1. The conceivable state space $S(x)$ and the lawful state space $S_L(x)$ of a thing x with two properties represented by functions p_1 and p_2. $R(p)$ is the range or set of values of p. Point s represents a state of thing x. (Redrawn from Bunge, 1977; reprinted by permission of Kluwer Academic Publishers.)

Natural kind (species)



As shown in this figure, P is a set of all properties, P(x) represents the properties of individual x, and P(y) represents the properties of individual y, L represents all the laws. According to this, x and y share the set of laws "L(x,y)", hence x and y are in the same natural kind (species).

Definitions of species

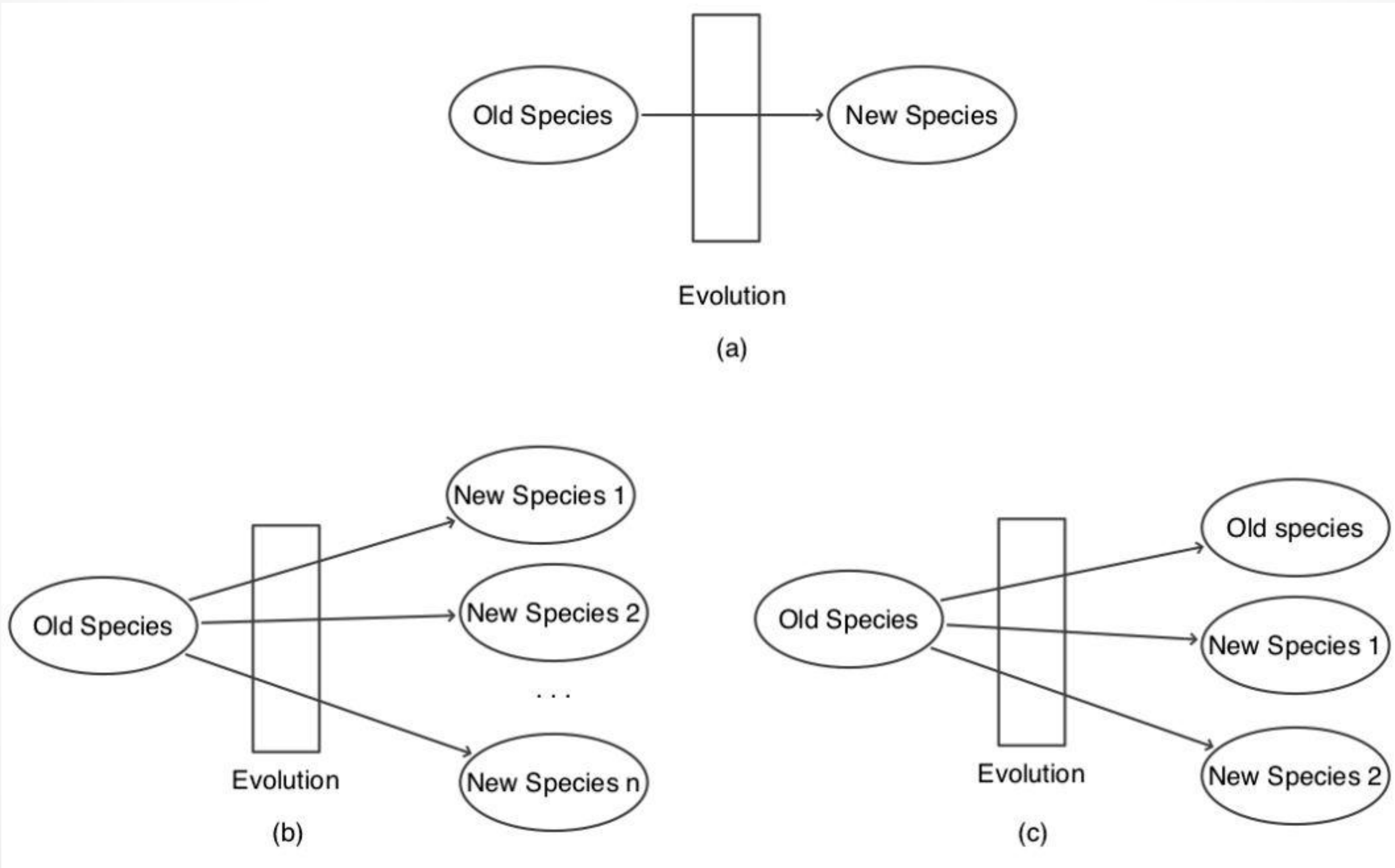
- Biological species

a) It is a natural kind (rather than an arbitrary collection),
b) All of its members are organisms (present, past, or future),
c) It "descends" from some other natural kind (biotic or prebiotic).

- Software species

a) It is a natural kind, an abstract class contain the laws constraining its members;
b) All of its members are copies of software;
c) The structure of all software species is like a forest but not a tree as bio-species, to count two elements in the same species, they have to be in the same tree.

Evolution situations



Evolution, Maintenance and Adaptation

	Happens at	Formulas
Evolution	Species level	$(D, S' \vdash R') \wedge (D_{Des}, Des' \vdash S') \wedge (D_I, I' \vdash Des')$ $(D', S' \vdash R) \wedge (D'_{Des}, Des' \vdash S') \wedge (D'_I, I' \vdash Des')$
Maintenance	Individual level	$(D, S \vdash R) \wedge (D_{Des}, Des \vdash S) \wedge (D_I, I' \vdash Des)$ $(D, S \vdash R) \wedge (D_{Des}, Des' \vdash S) \wedge (D_I, I' \vdash Des')$
Adaptation	Individual level	$(D, S \vdash R) \wedge (D_{Des}, Des \vdash S) \wedge (D_I, I' \vdash Des)$ $(D, S \vdash R) \wedge (D_{Des}, Des' \vdash S) \wedge (D_I, I' \vdash Des')$ $(D, S \vdash R) \wedge (D_{Des}, Des \vdash S) \wedge (D_I, I \vdash Des)$

Conclusion

- This paper aims at providing an ontology of software evolution
- Our work is mainly base on DOLCE framework
- Our work can be served as groundwork supporting other researches in software evolution.

Future work

- Firstly, more relating concepts should be present.
- Then, besides positioning the concepts into DOLCE framework, a set of formal constraints of these concepts should be provided.
- Finally, we need to adapt our ontology into real case studies to check its efficiency.

The end

Thanks!

References

- 1. Swanson, E.B., *The dimensions of maintenance*, in *Proceedings of the 2nd international conference on Software engineering*1976, IEEE Computer Society Press: San Francisco, California, United States. p. 492-497.
- 2. Godfrey, M.W. and D.M. German. *The past, present, and future of software evolution*. in *Frontiers of Software Maintenance, 2008. FoSM 2008*. 2008.
- 3. Vieu, L., S. Borgo, and C. Masolo, *Artefacts and Roles: Modelling Strategies in a Multiplicative Ontology*, in *Proceedings of the 2008 conference on Formal Ontology in Information Systems: Proceedings of the Fifth International Conference (FOIS 2008)*2008, IOS Press. p. 121-134.
- 4. Masolo, C., et al., *WonderWeb Deliverable D18 Ontology Library (final)*, 2003.
- 5. Daniel Oberle, S.G., Steffen Staab, *An Ontology for Software*, in *Handbook on Ontologies*, R.S. Steffen Staab, Editor 2009, Springer.
- 6. Mahner, M., *What is a species?* Journal for General Philosophy of Science, 1993. **24**(1): p. 103-126.