An Ontology of Software Evolution

A Research Proposal

Xiaowei Wang

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

Outline

1. Context and Motivation
2. Research Problem
3. Research Approach
4. Related Work
5. Evaluation Plan
6. Conclusion

1. Context and Motivation

People rely on software:

e.g. bank, hospital, government…

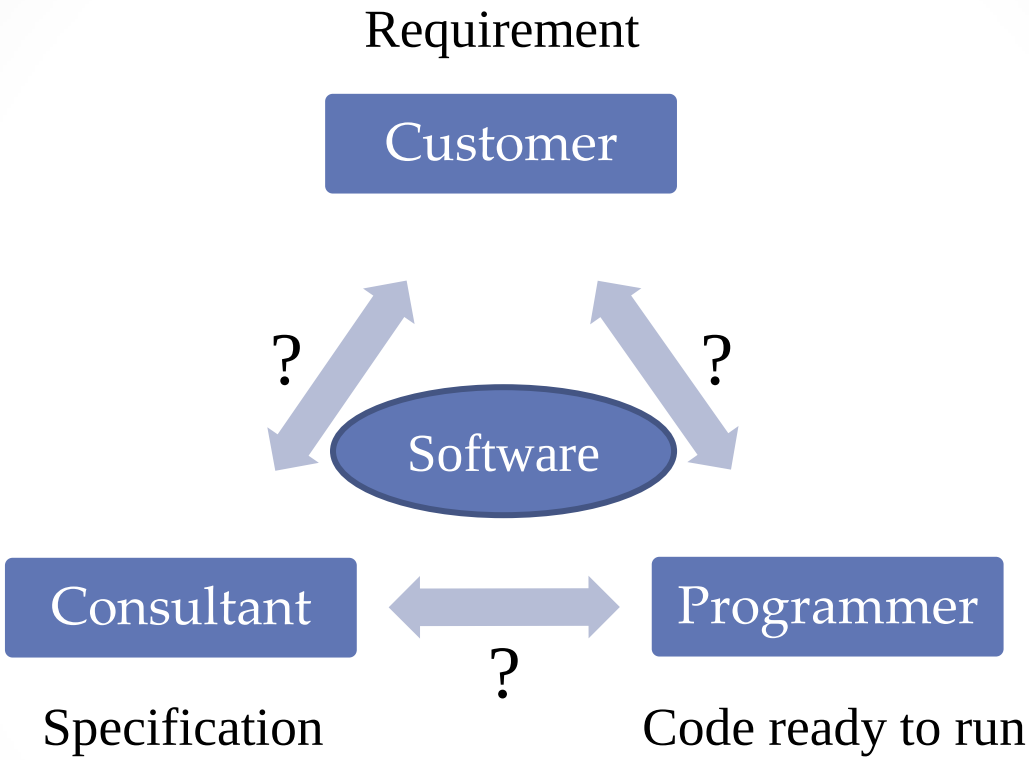
Software changes rapidly:

e.g. Windows xp-> vista-> 7

Managing changes is costly:

costs over 50% project budget

1.1 Concept Ambiguity



The concept ambiguity makes software developing a never ending iterating process

1.2 Knowledge Missing

Shortage in human memory

- Original developers may leave the project
- Current maintainers may forget the details

Shortage in documentation

- Few documents are available
- Documents are out of date

2. Research Problem

for concept ambiguity

- **2.1 Concept of software**
- **2.2 Concept of software evolution**

for knowledge missing

- **2.3 Methods and tools for software evolution**

2.1 concept of software

- Algorithm (e.g. a bubble sorting algorithm)
- Source code (e.g. encoded in Java/C)
- Realization of source code (e.g. the code stored on a hard disk)
- Running process of algorithm (e.g. sorting process running in a computer)

- Specification document?
- Design document?

To understand software evolution, a deeper understanding of software itself is necessary and essential.

2.2 concept of software evolution

e.g. software evolution v.s. software maintenance

- Used interchangeably [5]
- Maintenance subsumes evolution [2]
- Evolution subsumes maintenance [7]
- Terms as “change” or “aging” are used to avoid misinterpretations [4], [15]

To clarify the concepts relating to software evolution, thereby getting a clear understanding of software evolution phenomena.

2.3 methods and tools for software evolution

Tools are usually designed as file-based, and this limits the capability to track the semantics of the changes.

- e.g. CVS (Concurrent Versions System)

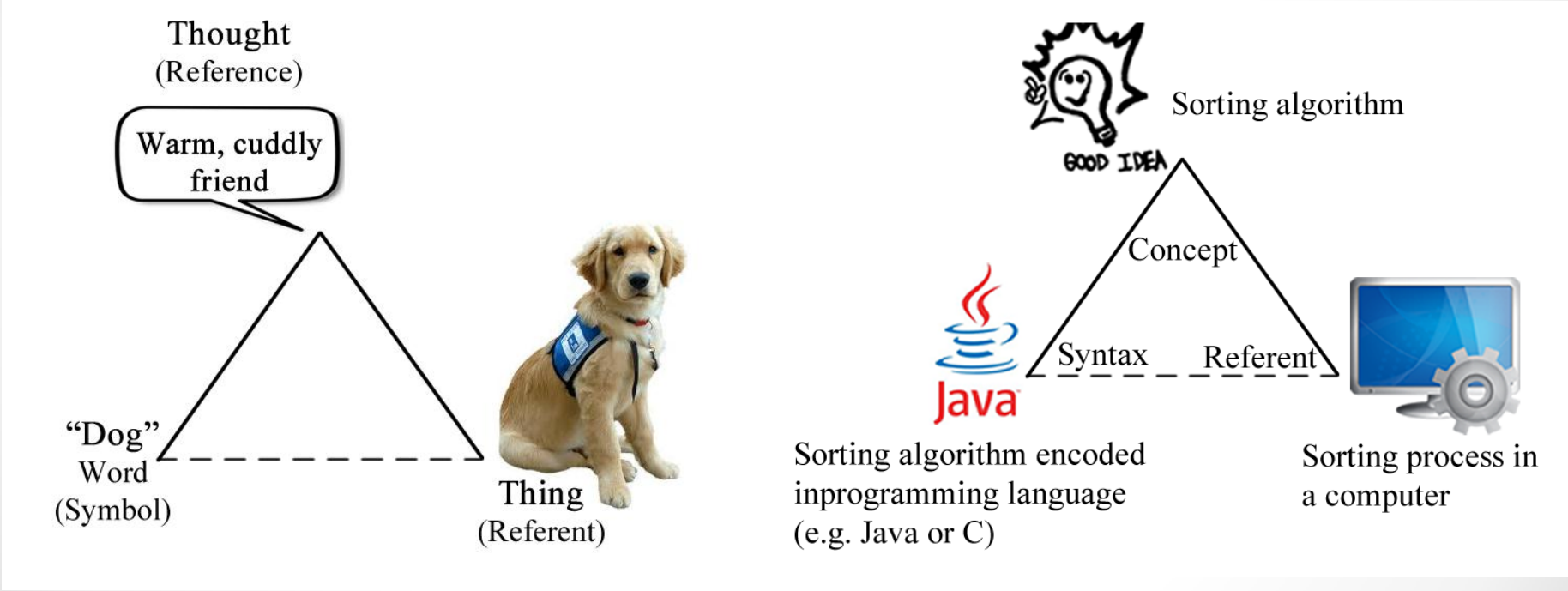
Dimension	CVS
Time of change	Compile-time
Change history	Any
Artifact	File
Granularity	File

To track the changes in software with higher semantics.

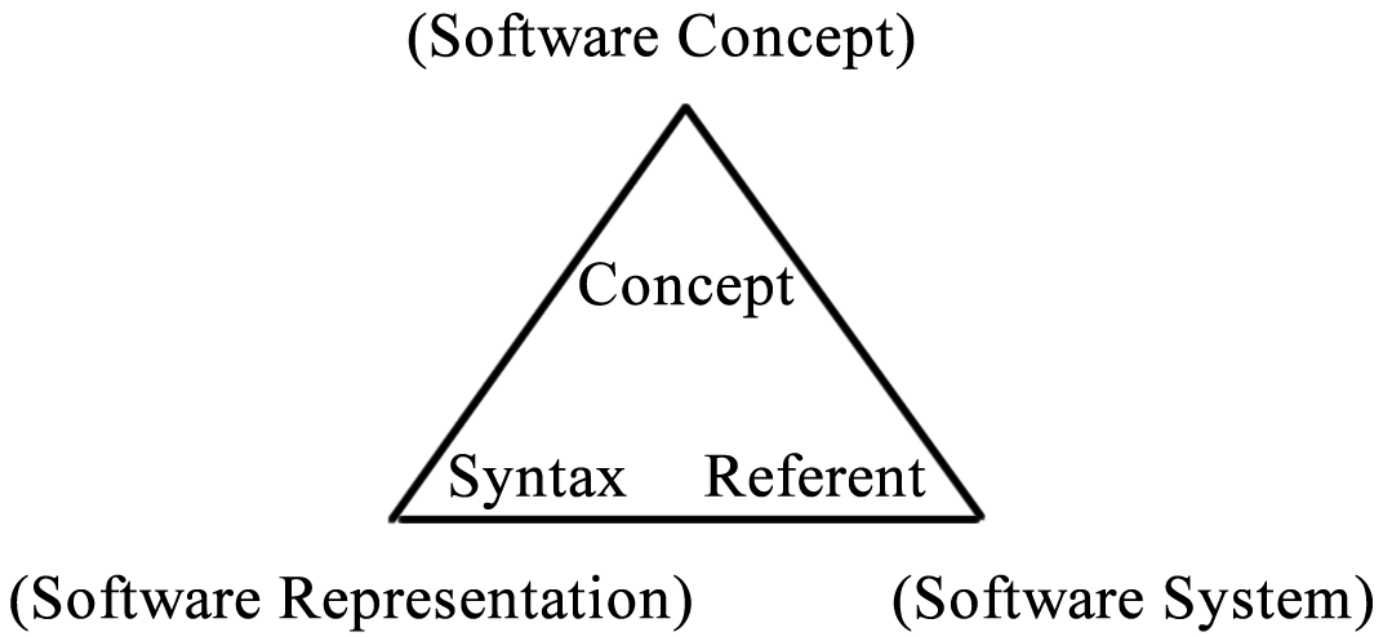
3. Research Approach

- **3.1 Ontology of software**
- **3.2 Ontology of software evolution**
- **3.3 Language for software evolution**

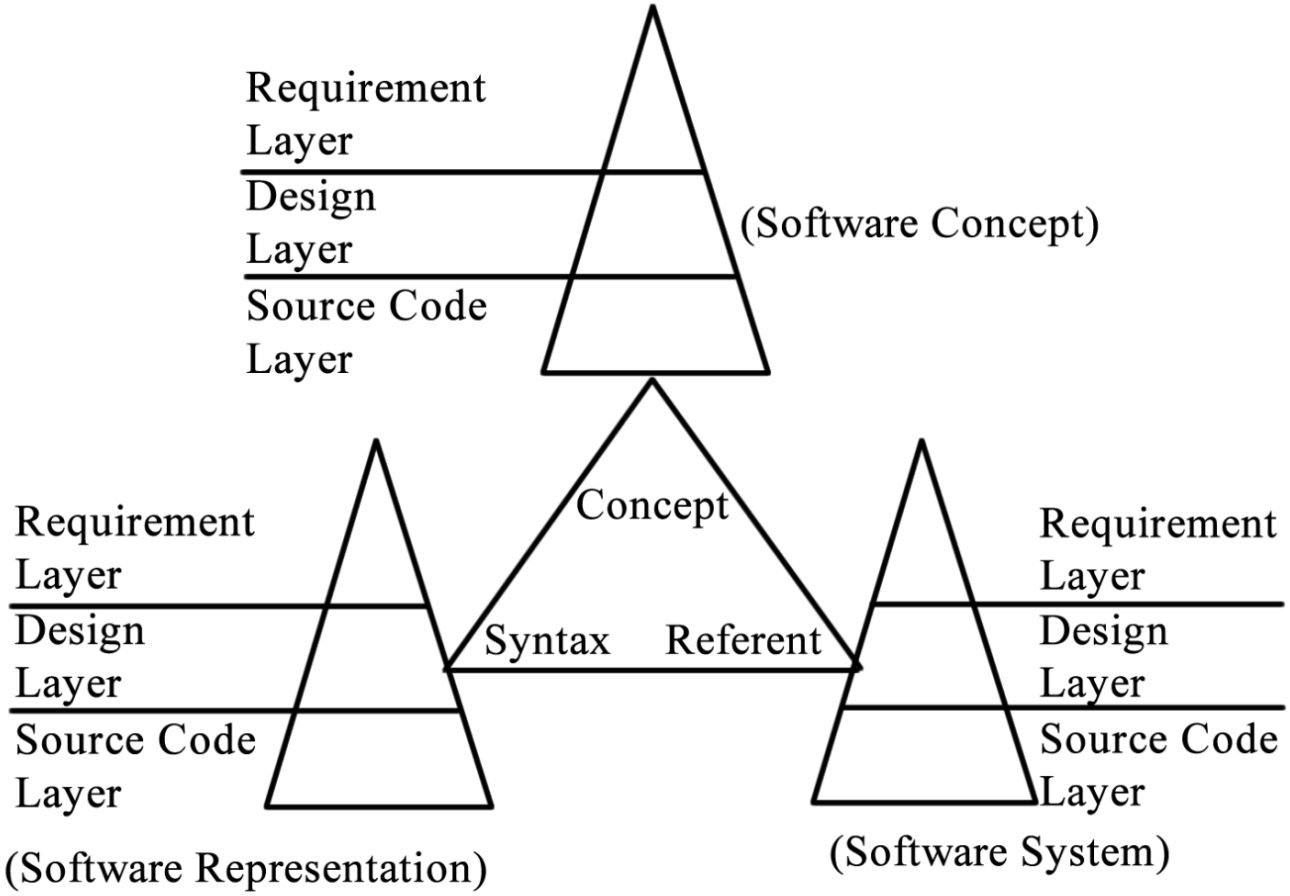
3.1 Ontology of software



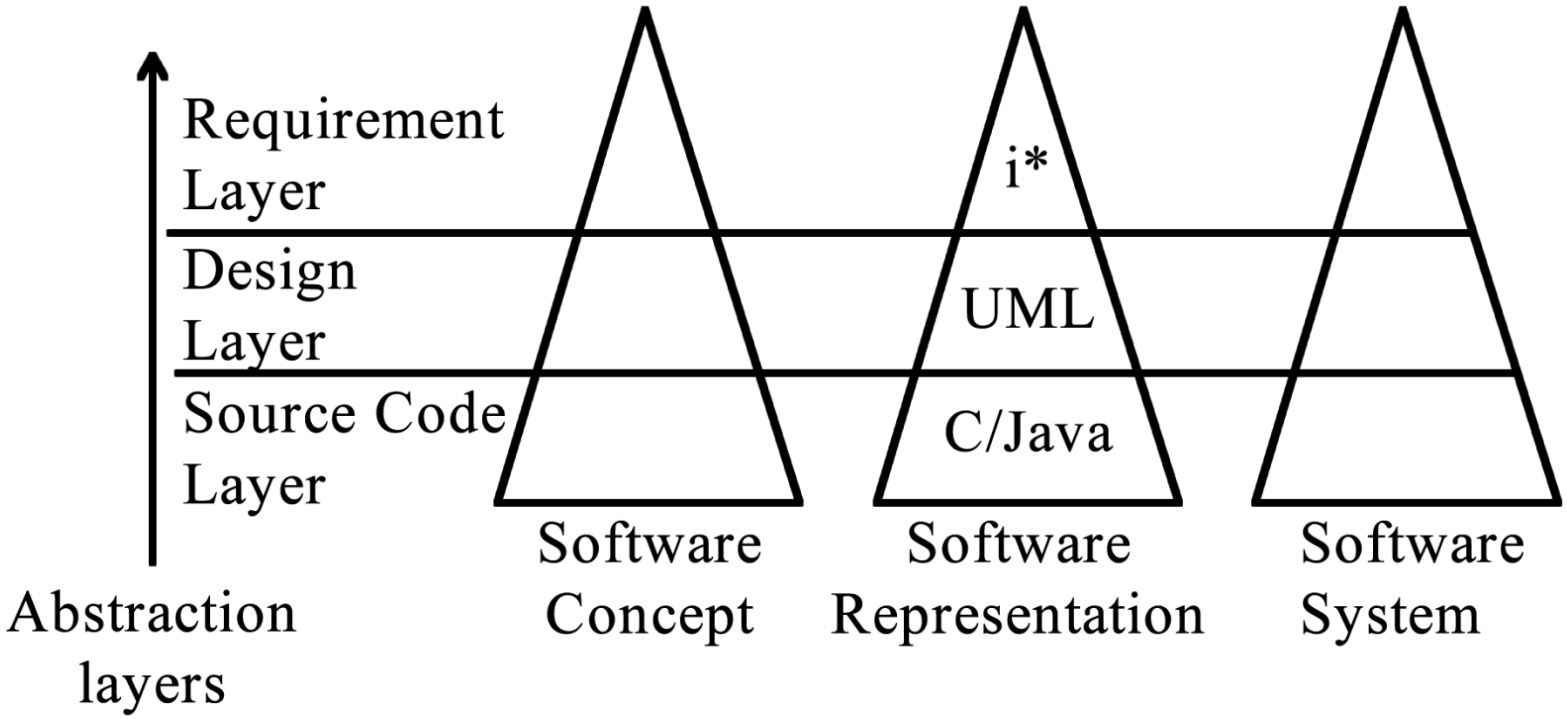
3.1 Ontology of software



3.1 Ontology of software



3.1 Ontology of software



3.1 Ontology of software

e.g. an email system

Software Representation (Syntax)	**Software System (Referent)**
A goal of managing emails	An email application
Design modules of email creating, receiving and sending	Activities of creating, receiving and sending emails
Source code fulfilling the design	The processes running in a computer

3.2 Ontology of software evolution

- Darwin published his memorable book *On The Origin of the Species*.
- Software and biological creatures are both living in the environments which are continuously changing.
- To survive in such continuously changing environments, software and biological creatures both need to change themselves to gain better adaptability.

3.2 Ontology of software evolution

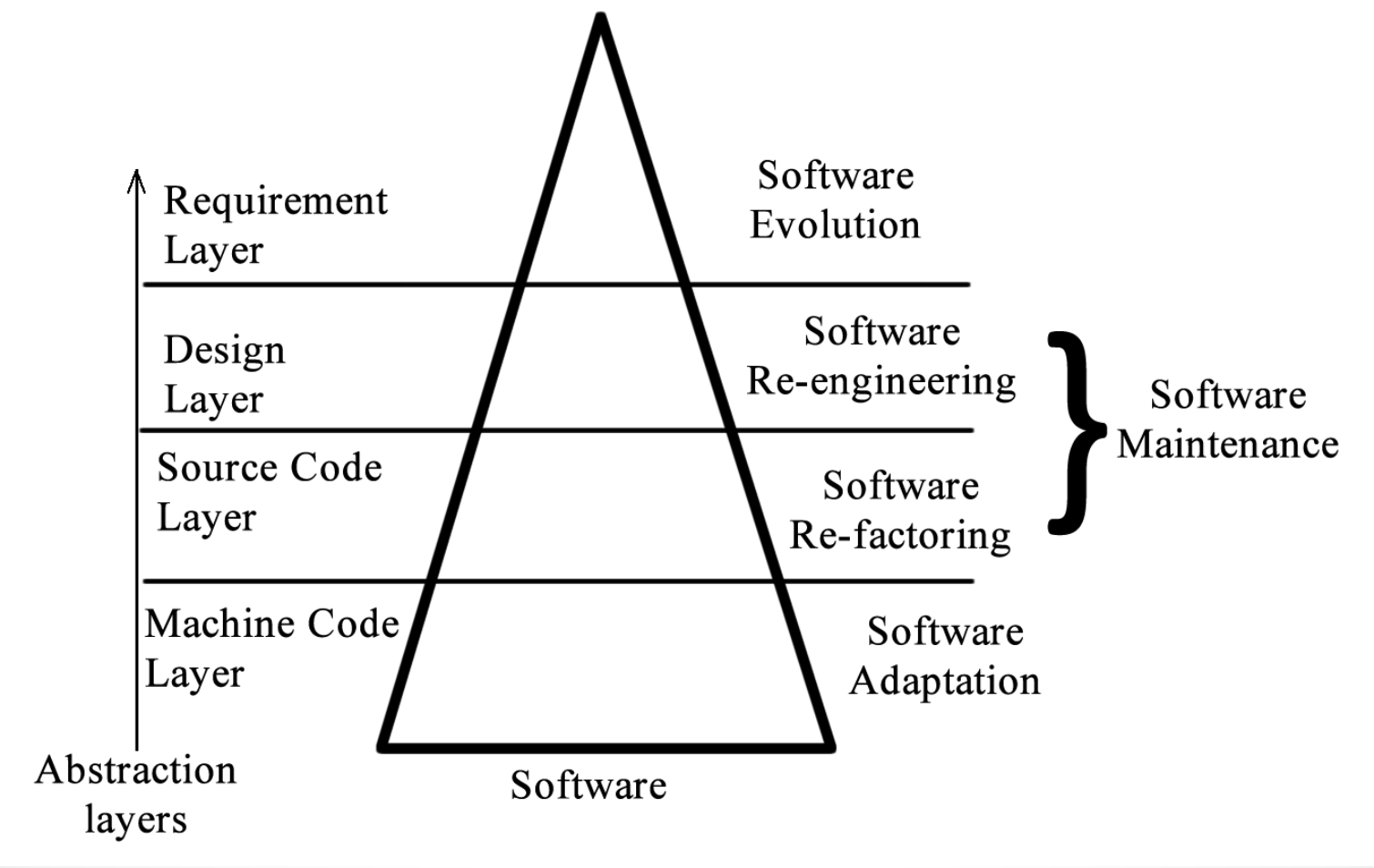
- Differently from living creature, software has no life.

- It is non-tangible and expressed through representation languages.

if gene is interpreted as “instructions of features”

- For living creatures, the gene (instruction) is stored in the cells (body), and the gene can be copied through cell reproduction.
- For software, the specification (instruction) and the source code (body) are stored separately, and the specification can not be copied through copying the source code.

3.2 Ontology of software evolution



3.2 Ontology of software evolution

e.g. versioning numbers

- Traditional versioning numbers are decided by the significance of changes between releases, but these changes are entirely arbitrary and up to the author.

- According to our software abstraction layers, the significance level might be determined as “v 1.2.3”
  - 1- Specification number
  - 2- Design number
  - 3- Source code number

3.3 Language for software evolution

$$(D, S \vdash R)$$

$$(D, S \vdash R) \wedge (Design \vdash S) \wedge (Source\ Code \vdash Design)$$

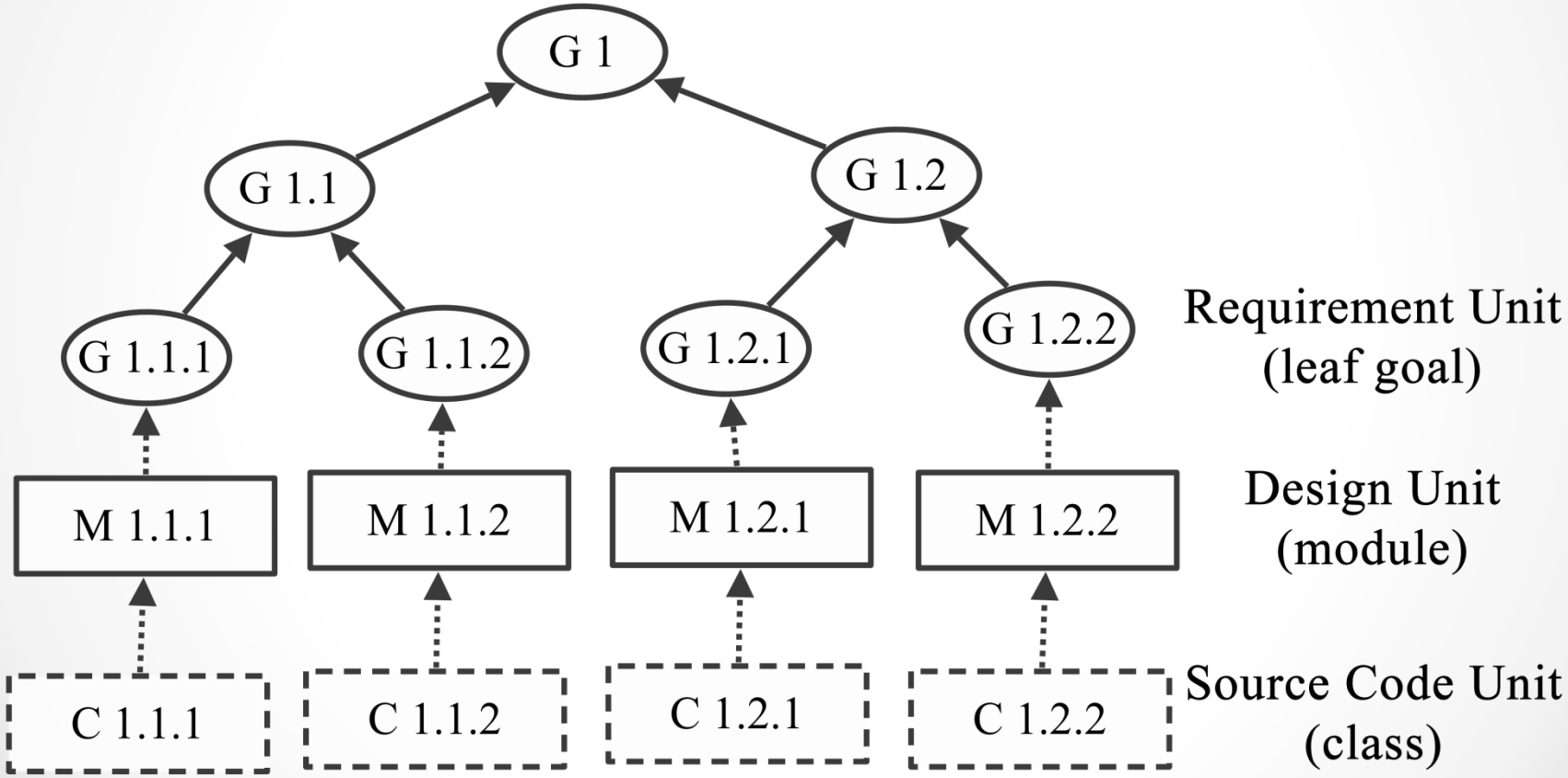
$$(D, S' \vdash R) \wedge (Design' \vdash S') \wedge (Source\ Code' \vdash Design')$$

3.3 Language for software evolution

- Levels of representation languages

Level	Primitive constructs	Interpretation
Logical	Predicates	Arbitrary
Epistemological	Structuring relations *concepts and roles*	Arbitrary
Ontological	Structuring relations *satisfying meaning postulates*	Constrained
Conceptual	Cognitive primitives	Subjective
Linguistic	linguistic primitives	Subjective

3.3 Language for software evolution



4. Related Work

- **4.1 Concept of software**
- **4.2 Software maintenance/evolution**
- **4.3 Methods and tools for software evolution**

4.1 Concept of software

Definition of software

- Osterweil [13], Eden [6], Martin [11]

Ontology of software

- Oberle [12], SWORD

Ontology of information object

- METOKIS, IAO

4.2 Software maintenance/evolution

Study in software engineering

- laws of software evolution, software process models, software configuration management, reverse engineering, refactoring

Metaphor between software and biological evolution

- Mahner [10], Godfrey [7]

Taxonomies of software evolution

- Lientz & Swanson [17], Chaptin [5], Buckley [4]

Ontologies of software evolution

- Kitchenham [9], Ruiz [16], Anquetil [1], Tappolet [19]

4.3 Methods and tools for software evolution

Software documentation

- Parnas [14], [15]

Tools and methods

- Buckley [4], Tang [18], Tappolet [19], Beyer [3]

Language extension based on ontology

- Giancarlo [8]

5. Evaluation Plan

OntoClean methodology

- Imposing several constraints on the taxonomic structure of an ontology, which could help in eliminating inappropriate and inconsistent modeling choices.

Prototype

- Provide a tool with the ontology-based language embedded
- Adapt this tool in a software developing project
- Collect the feedback from stakeholders

6. Conclusion

- This project aims at providing an ontology of software, an ontology of software evolution, and a ontology-base language
- We try to get a deeper understanding of software evolution phenomena, thereby facilitating the difficulty in software evolution.
- We hope our work could be served as groundwork supporting other researches in software evolution.

The end

Thanks!

References:

- [1] Anquetil, N. et al. 2007. Software maintenance seen as a knowledge management issue. *Information and Software Technology*. 49, 5 (May. 2007), 515–529.
- [2] Bennett, K.H. and Rajlich, V.T. 2000. Software maintenance and evolution: a roadmap. *Proceedings of the Conference on The Future of Software Engineering* (New York, NY, USA, 2000), 73–87, DOI= 10.1145/336512.336534.
- [3] Beyer, D. and Hassan, A.E. 2006. However, version control systems (VCS) contain valuable historical information about a project, and mining the VCS repository may reveal interesting events in the development and maintenance of long-lived projects. *Reverse Engineering, 2006. WCRE '06. 13th Working Conference on*, DOI= 10.1109/WCRE.2006.14.
- [4] Buckley, J. et al. 2005. Towards a taxonomy of software change: Research Articles. *J. Softw. Maint. Evol.* 17, 5 (2005), 309–332, DOI= 10.1002/smr.v17:5.
- [5] Chapin, N. et al. 2001. Types of software evolution and software maintenance. *Journal of Software Maintenance*. 13, 1 (2001), 3–30.
- [6] Eden, A.H. and Turner, R. 2007. Problems in the ontology of computer programs. *Appl. Ontol.* 2, 1 (2007), 13–36.

References:

- [7] Godfrey, M.W. and German, D.M. 2008. The past, present, and future of software evolution. *Frontiers of Software Maintenance, 2008. FoSM 2008.*, DOI= 10.1109/FOSM.2008.4659256.
- [8] Guizzardi, G. 2005. *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology.
- [9] Kitchenham, B.A. et al. 1999. Towards an Ontology of software maintenance. *Journal of Software Maintenance*. 11, 6 (1999), 365–389, DOI= 10.1002/(SICI)1096-908X(199911/12)11:6<365::AID-SMR200>3.0.CO;2-W.
- [10] Mahner, M. 1993. What is a species? *Journal for General Philosophy of Science*. 24, 1 (1993), 103–126, DOI= 10.1007/bf00769517.
- [11] Martin, J. 2010. *Introduction to Languages and the Theory of Computation*. McGraw-Hill Companies,Incorporated.
- [12] Oberle, D. et al. 2009. An Ontology for Software. S. Staab and D. Rudi Studer, eds. Springer Berlin Heidelberg. 383–402, DOI= 10.1007/978-3-540-92673-3_17.
- [13] Osterweil, L.J. 2008. What is software? *Autom. Softw. Eng.* 15, 3-4 (2008), 261–273.

References:

- [14] Parnas, D.L. 2011. Precise Documentation: The Key to Better Software. S. Nanz, ed. Springer Berlin Heidelberg. 125–148, DOI= 10.1007/978-3-642-15187-3_8.
- [15] Parnas, D.L. 1994. Software aging. *Proceedings of the 16th international conference on Software engineering* (Los Alamitos, CA, USA, 1994), 279–287.
- [16] RUIZ, F. et al. 2004. An ontology for the management of software maintenance projects. *International Journal of Software Engineering and Knowledge Engineering*. 14, 03 (Jun. 2004), 323–349, DOI= 10.1142/S0218194004001646.
- [17] Swanson, E.B. 1976. The dimensions of maintenance. *Proceedings of the 2nd international conference on Software engineering* (Los Alamitos, CA, USA, 1976), 492–497.
- [18] Tang, A. et al. 2011. Software Architecture Documentation: The Road Ahead. *Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on*, DOI= 10.1109/WICSA.2011.40.
- [19] Tappolet, J. et al. 2010. Semantic web enabled software analysis. *Web Semant.* 8, 2-3 (2010), 225–240, DOI= 10.1016/j.websem.2010.04.009.