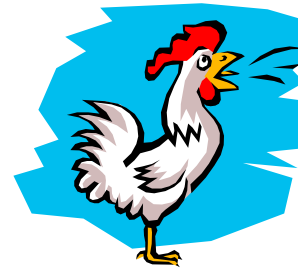


Development/Maintenance/Reuse: Software Evolution in Product Lines



Prof. Stephen R. Schach

Vanderbilt University, Nashville, TN

Dr. Amir Tomer

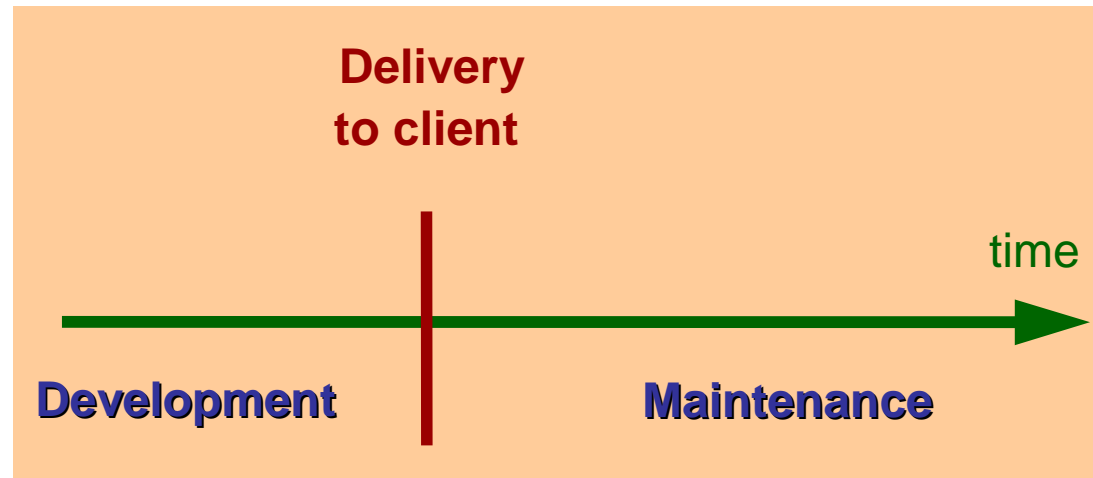
Rafael Ltd., Haifa, Israel

Overview

- **Development vs. Maintenance**
- **Evolution of a software product**
 - **The Evolution Tree Model**
- **Software Product Lines & S/W Reuse**
 - **A 3-dimensional Model for Product Line Evolution**
- **Conclusions and references**

What is Software Maintenance (1)?

- **IEEE 610.12 (1990) Definition:**
 - “The process of modifying a software system or component after delivery...”

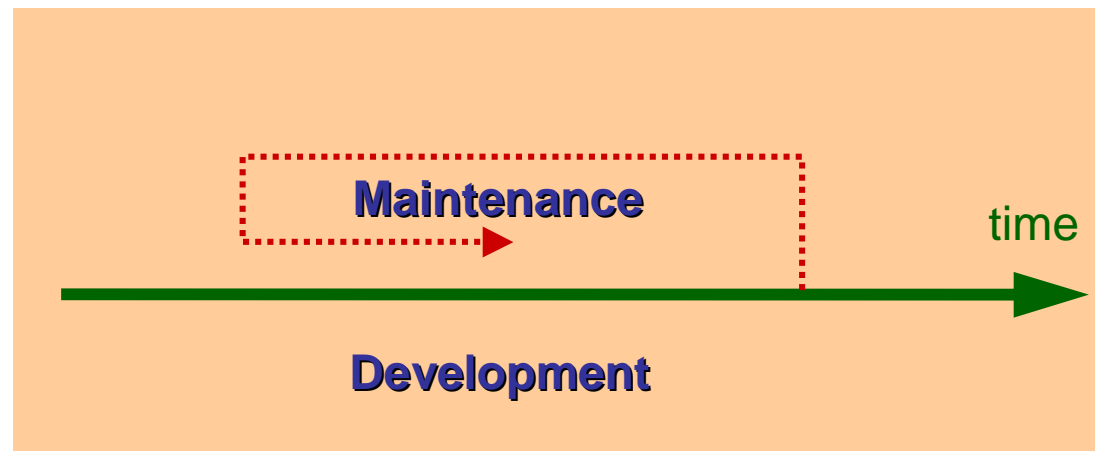


- **Is there any difference between pre-delivery debugging and after-delivery debugging?**

IEEE 610.12 definition is temporal, not operational

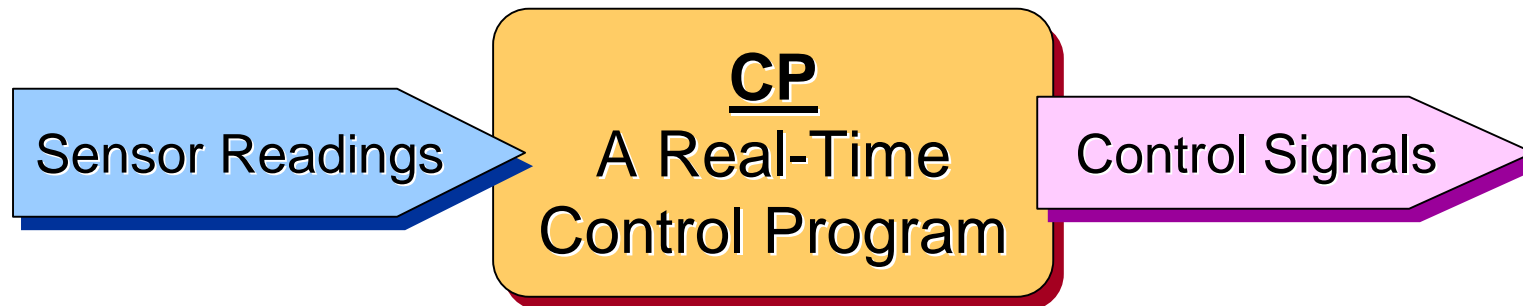
What is Software Maintenance (2)?

- **IEEE/EIA 12207 (1995) Definition:**
 - “The Maintenance Process... is activated when the software product undergoes modifications to code and associated documents...”

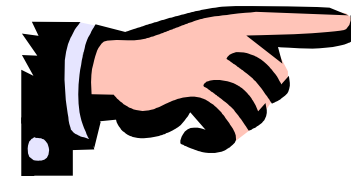


- **Is there any difference between development and maintenance?**

Evolution of a Software Product: The Story of CP



- **Prologue**
 - Software development according to requirements
- **Episode 1**
 - Problem: software does not meet R/T performance
 - Cause: The compiler generated overhead machine code
 - Solution: change variable types to avoid type-casting
 - Result: recoding



On with the Story...



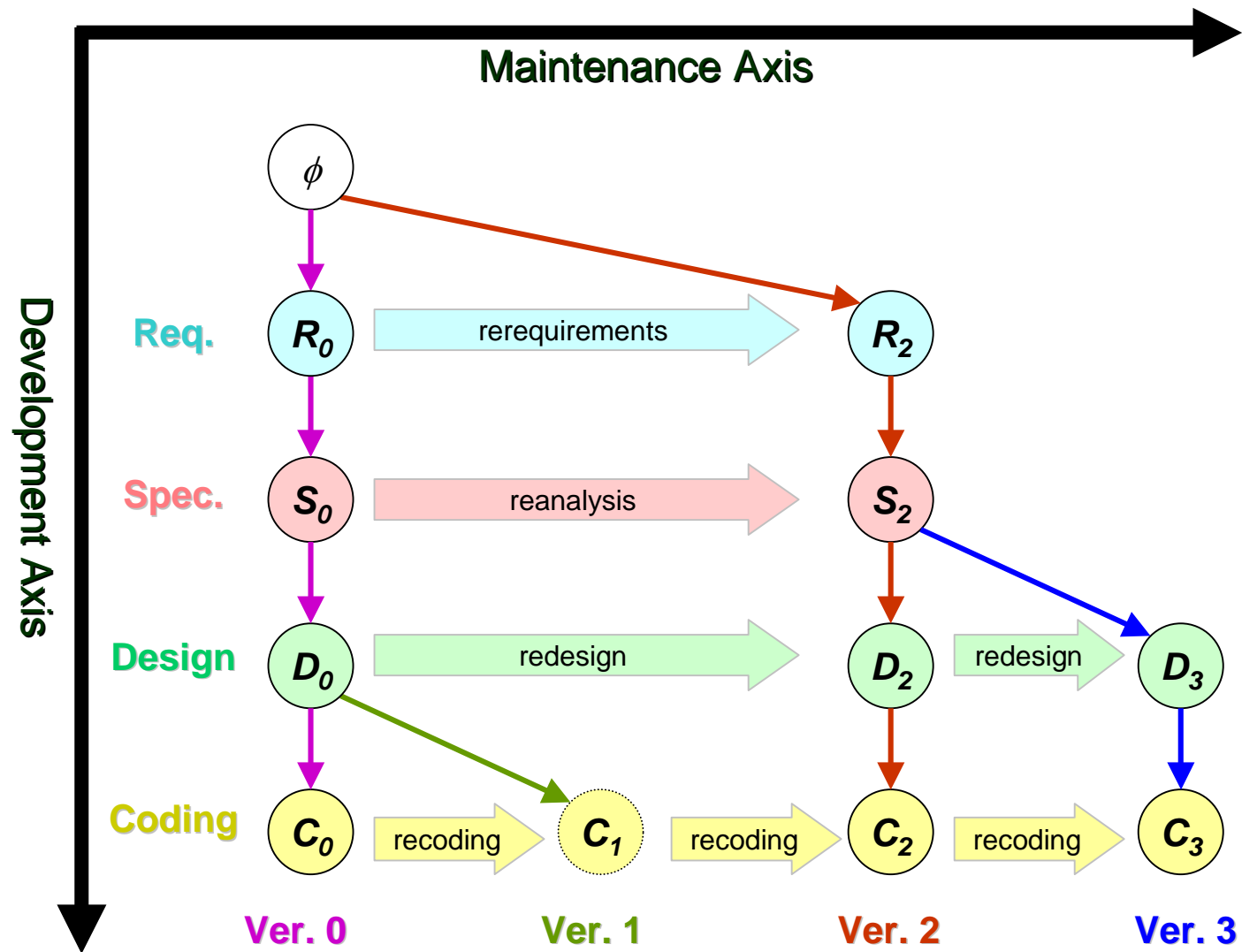
- **Episode 2**

- Problem: software does not meet R/T performance
- Cause: The actual sensors obtained have different specifications
- Solution: replace trigonometric library functions by pre-computed data tables
- Result: rerequirements, reanalysis, redesign and recoding

- **Episode 3**

- Problem: The controller is not adequate as an off-the-shelf-product (a marketing-based requirement)
- Cause: Performance is not compatible with a wide range of commercial sensors
- Solution: Enhance performance as much as possible within existing specification.
- Result: redesign and recoding

The Evolution Tree of CP



Product Lines and Reuse

- **Software Product line:**

- "A set of software-intensive products... sharing a common, managed set of features"

- Clements & Northrop – A framework for software product line practice

- **Reuse**

- Black-box reuse
- Glass-box reuse
- COTS

- **Reusable artifacts**

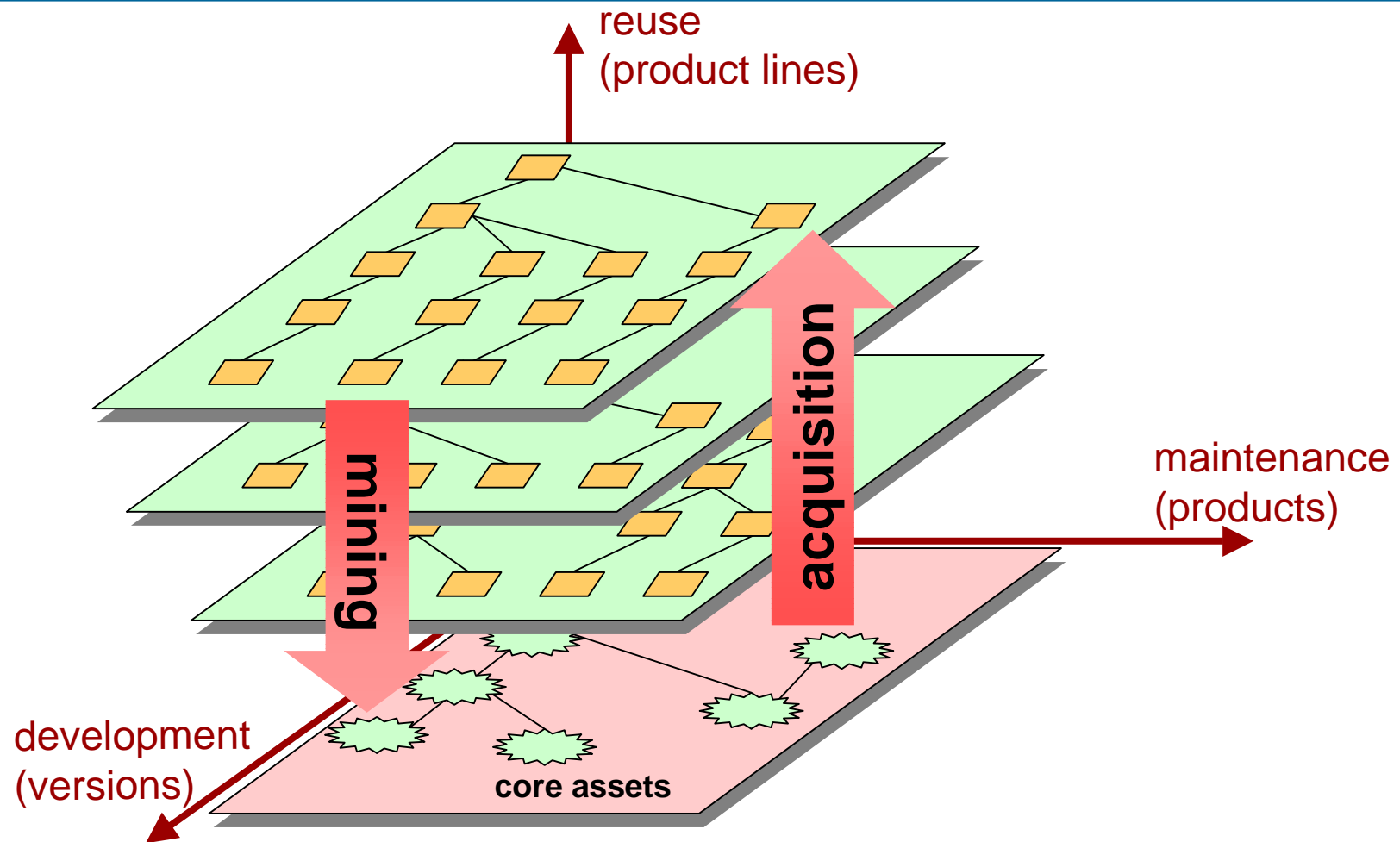
- Documents
- Class specifications
- Design patterns
- Algorithms
- Code modules
- ...

Each product in the product-line has its own evolution tree

Reusable artifacts transfer between products

An asset repository may be utilized to manage all reusable artifacts

Three Dimensional Software Engineering**

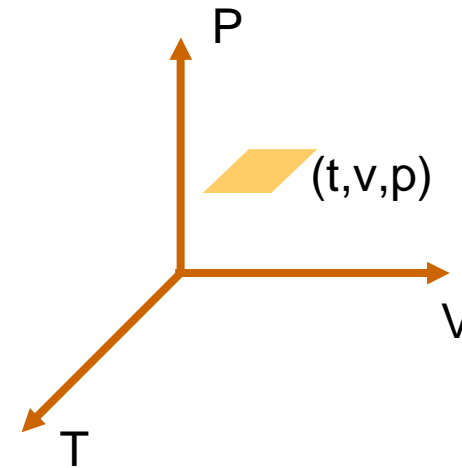


**Intl. Conf. on S/W Product Lines, August 2000 (to appear)

Artifact Location in the 3D Evolution Space

Each individual artifact is located in the 3D software evolution space. Its "coordinate" is denoted by a triple (T,V,P) , as follows:

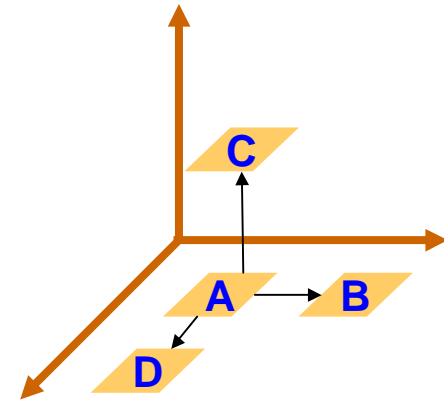
- **T = Type**
 - The location along the development axis
 - Types should be ordered
 - Other type examples:
 - Inception/elaboration/construction/transition (RUP)
 - Use-cases/sequence-diagrams/code-modules (UML)
- **V = Version**
 - The location along the maintenance axis
 - Usually a decimal number
 - Specifies the time ordering of artifacts of the same type
- **P = Product**
 - The location along the reuse axis
 - Specifies the identification of the specific product library in which the artifact resides



Example: ('Target class', 3.7, 'RS for navy')

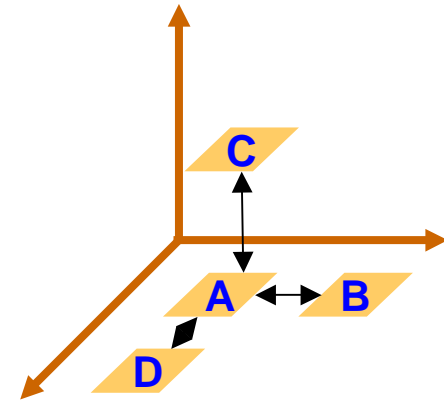
The Three Basic Operations of Software Evolution

- **B = modify(A)**
 - Artifact B is obtained by making modifications to artifact A
 - A "move" along the maintenance axis.
 - $(t, v', p) = \text{modify}(t, v, p)$
- **C = copy(A)**
 - Artifact C is an exact replica of artifact A
 - A "move" along the reuse axis
 - $(t, v, p') = \text{copy}(t, v, p)$
- **D = generate(A)**
 - An artifact D is generated from another artifact A.
 - Example: A class diagram is generated from the requirements document
 - A "move" along the development axis
 - $(t', v, p) = \text{generate}(t, v, p)$



“Single Step” Software Activities

- **Engineering**
 - $D = \text{generate}(A)$
 - Example: design derived from specification
- **Reverse Engineering**
 - $A = \text{generate}^{-1}(D)$
 - Example: design derived from code
- **Reengineering**
 - $B = \text{modify}(A)$
 - Example: add a relation to a class diagram
- **Rollback**
 - $A = \text{modify}^{-1}(B)$
 - Example: undo the changes
- **Acquisition**
 - $C = \text{copy}(A)$
 - Example: a design pattern is acquired from the core asset repository to a specific product
- **Mining**
 - $C = \text{copy}^{-1}(A)$
 - Example: a code module is extracted from a specific product to the core asset repository



Combined Software Reuse Activities

- **White-box Reuse**
 - $P = \text{modify}(\text{copy}(C))$
 - P is a specific design implementing a core design pattern C taken from the repository
- **Infrastructure Modification**
 - $P = \text{copy}(\text{modify}(C))$
 - A modified core asset [$\text{modify}(C)$] is prepared to be acquired by several products using P .
- **Asset “standartization”**
 - $C = \text{modify}(\text{copy}(P))$
 - A specific component P is mined from a product and then transformed into a standard form (e.g. COM) before being stored in the core asset repository
- **Re-mining modified assets**
 - $C = \text{copy}(\text{modify}(P))$
 - A component P , which was previously mined as a core asset, has been modified in the product, and its newer version is now being re-mined.

Recording Evolution History

- **Using the 3 basic operators and their reverse forms, the evolution history of each artifact may be tracked**
- **The evolution history may be embedded within the artifact itself**
- **The evolution history may help to configure new products in the product line based on artifacts from different “ages”.**

Conclusions

- **Development, Maintenance and Reuse may be described in a unified evolution model**
- **The entire evolution of a software product line may be described in a three-dimensional space**
- **Every artifact (either actual or historical) is "located" in the 3D evolution space**
- **The 3D software evolution model provides a means for controlling artifacts throughout entire life cycle**

References

- **The Evolution Tree**
 - Tomer, A and Schach, S. R., “The Evolution Tree: A Maintenance-Oriented Software Development Model”, *Proceedings of the Fourth European Conference on Software Maintenance and Reengineering (CSMR'2000)*, Zurich, Switzerland, February/March 2000, pp. 209-214.
- **The Propagation Graph**
 - Schach, S. R. and Tomer, A., A Maintenance-Oriented Approach to Software Construction, *Journal of Software Maintenance: Research and Practice*, vol. 12, February, 2000, pp. 25-45.
- **The 3-dimensional ET model**
 - Schach, S. R. and Tomer, A., “Development/Maintenance/Reuse: Software Evolution in Product Lines”, *Proceedings of the First Software Product Lines Conference (SPLC1)*, Denver, USA, August, 2000, pp. 437-450.
- **A 3D evolution model for system design**
 - Tomer, A and Schach, S. R., “A Three-Dimensional Model for System Design Evolution”, to appear in *Journal of Systems Engineering*, 2002

Contacts

