CSC 408F/CSC2105F Lecture Notes

These lecture notes are provided for the personal use of students taking CSC 408H/CSC 2105H in the Fall term 2004/2005 at the University of Toronto.

Quality Matters

- Producing quality products has been identified as a key factor in the long term success (i.e. profitability) of organizations.
- Quality doesn’t happen by chance.
  Quality requires on ongoing commitment by all parties (management, designers, developers).
- The commitment to quality includes the use of careful development processes. Quality control must be embedded into the process. [The quality movement in Japan. Process quality ...]

Managing Software Quality

- Define what *quality* means for large software systems
- Measure Quality of a complete or partial system.
- Devise actions to improve quality of the software
  - Process improvements [Process Performance improvements $\Rightarrow$ Product Productivity improvements]
  - Product improvements
- Monitor Quality during development
- *Software Quality Assurance* - a team devoted to encouraging and enforcing quality standards.

Defining Software Quality

- Software quality is defined as

  1. Conformance to explicitly stated functional [correctness] and non-functional requirements [performance, security, maintanability, usability, etc.]
     i.e. Build the software described in the system Requirements and Specifications
  2. Conformance to explicitly documented development standards
     i.e. Build the software the right way
  3. Conformance to implicit characteristics that are expected of all professionally developed software
     i.e. Build software that meets the expectations of a reasonable person
     in law this is called the principle of merchantability

- Assessing software quality is often based on *subjective* criteria.
  As yet there is no generally accepted computable metric for software quality.

- Although it has proven difficult to quantify and measure, it is often easy to identify real quality [qualitatively].

Quality Metrics

There have been some efforts to develop numerical metrics for software quality.

- Correctness
  - Formal proof or program verification
  - Unit test, Integration test, System test, Acceptance test
- Reliability
  - Mean Time Between Failures

- Complexity
  - Source size or compiled size.
    - Lines of code (LOC)
    - McCabe's complexity:

      $$|V| + |E| - 2$$

      for a control flow graph G=(V, E).
    - Halstead's Software Science $\Leftarrow$ Information Science

      $$(N_1 + N_2)\log(n_1 + n_2)$$
  - OO Software Metrics
    - Cohesion metrics in Packages, Classes, Methods
    - Coupling metrics in Packages, Classes, Methods

- Performance
  - Time, in relation to the input size
    - CPU cycles, in relation to the input size
    - Cache misses, in relation to the input size
    - Network delay, system perf.
    - Network throughput, system perf.
  - Space, in relation to the input size
    - Workload, in relation to the input size
    - Network traffic, in relation to the input size

- Usability
  - Operational usability
  - Memorizing usability
  - Manual completeness
- Security
  - Types of vulnerabilities
  - Number of vulnerabilities
  - Encryption strength [MD5 story]

Quality Tradeoffs

References:

Y. Yu et al. ”Software refactorings guided by soft-goales”. 1st Refactorings workshop, WCRE, 2003.

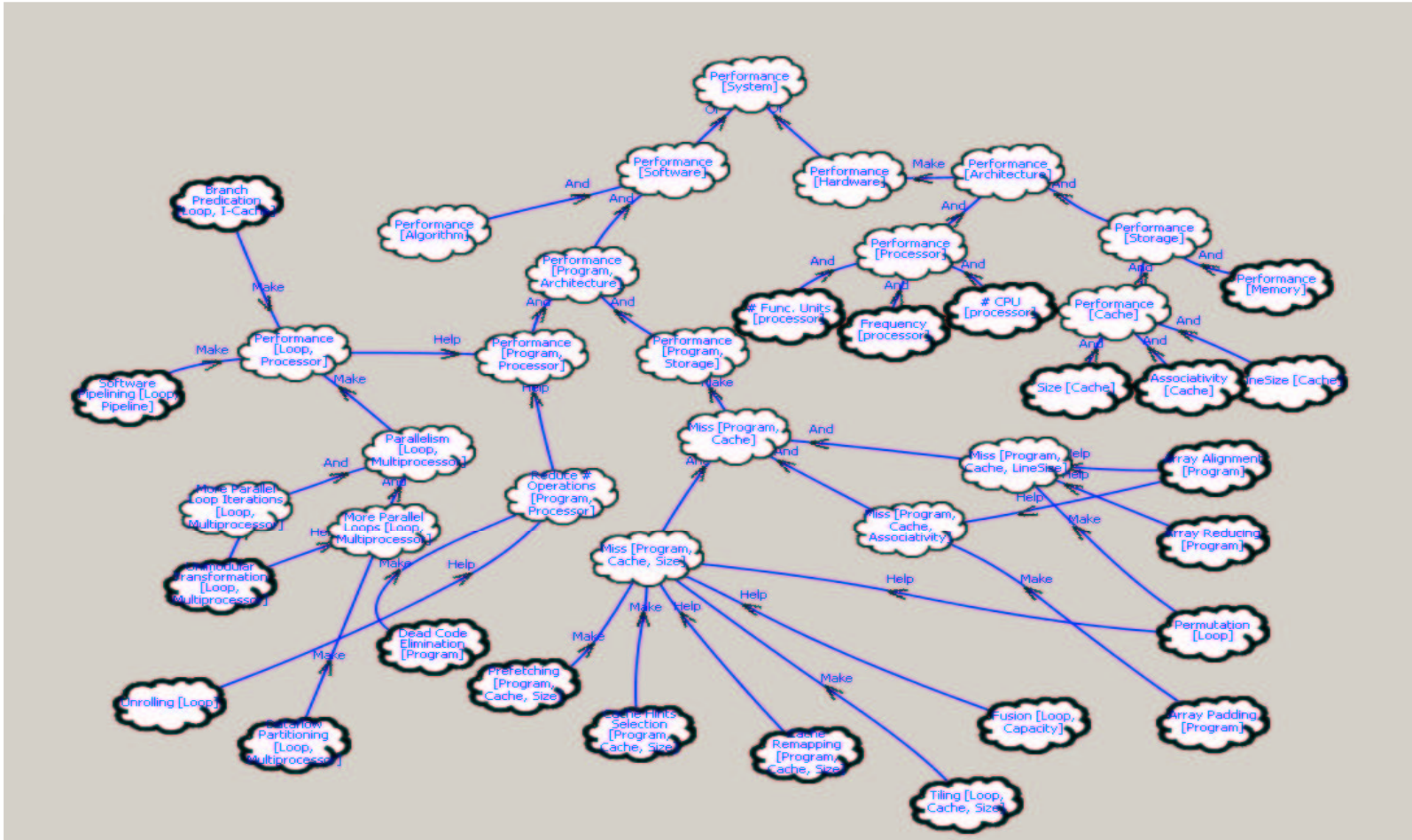
For the quality requirements modelling tool, see:
http://www.cs.toronto.edu/ yijun/OpenOME.html

[Question: FORTRAN stands for ... ?]

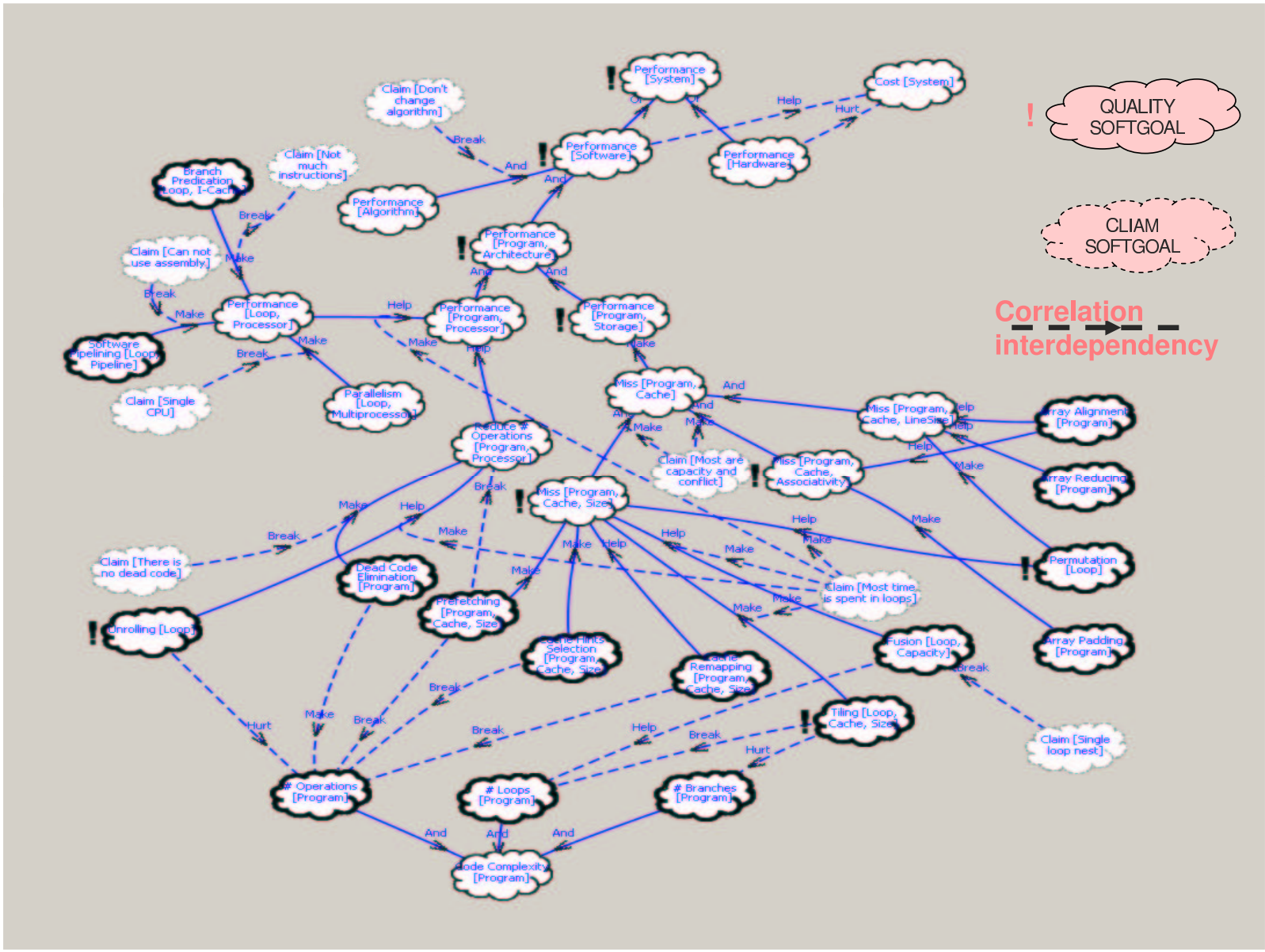
Example. Matrix Multiplication

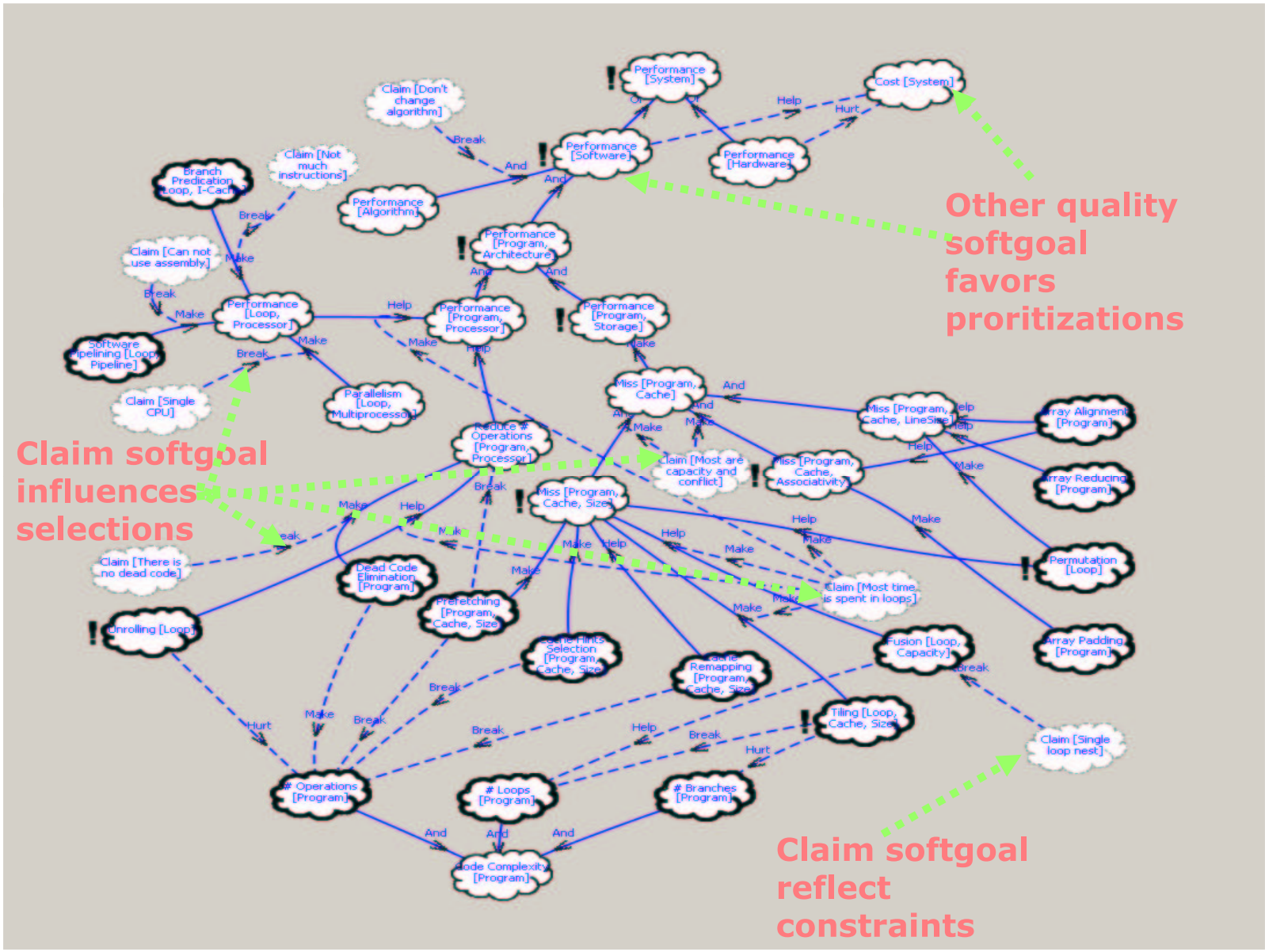
```
real*8 A(512,512),B(512,512),C(512,512)
do i = 1 , M
  do j = 1, L
    do k = 1, N
      C(i,k) = C(i,k) + A(i,j) * B(j,k)
```

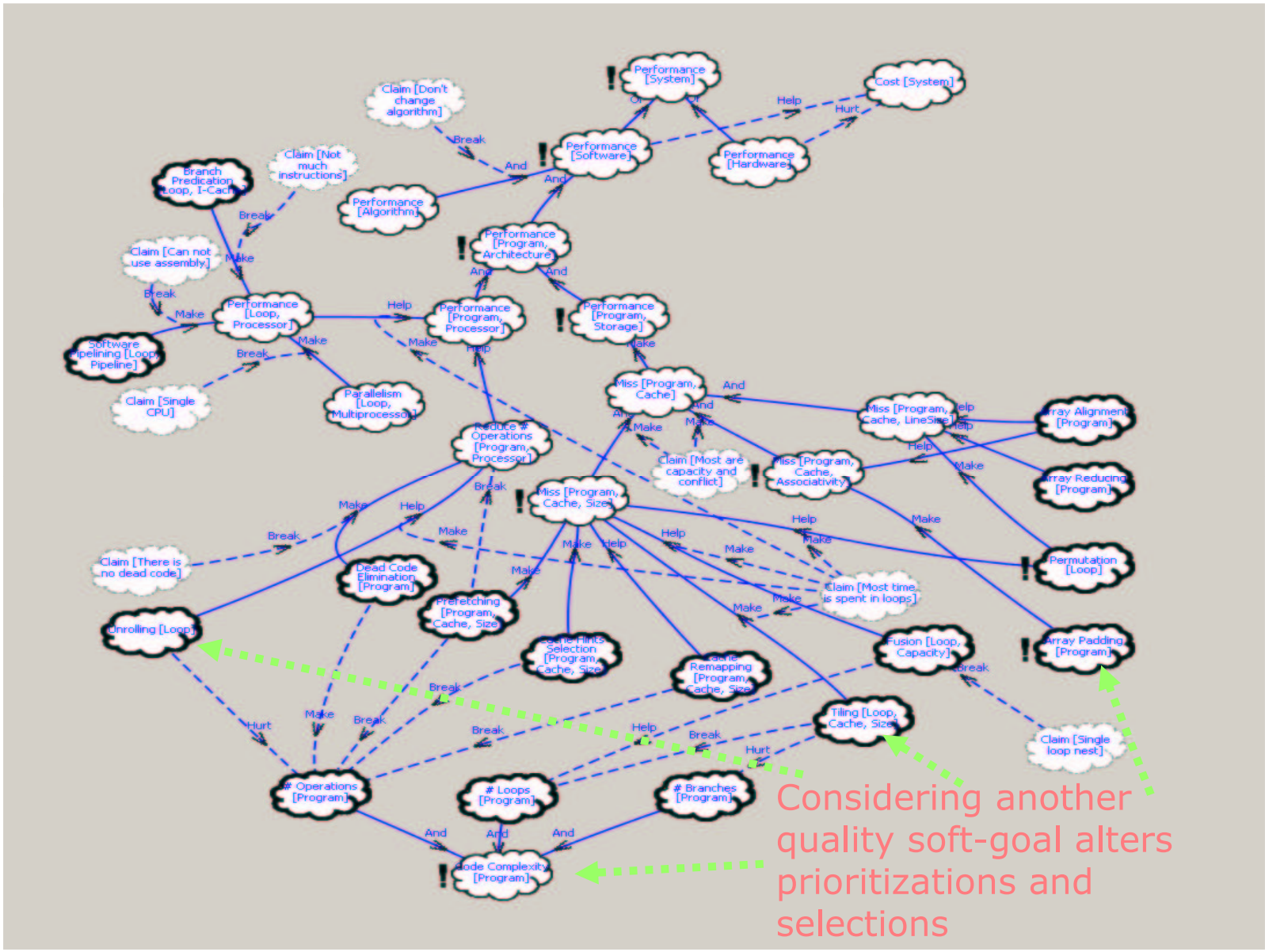
Quality goal: ”speedup the program 20 times without sacrificing the code complexity 4 times”.

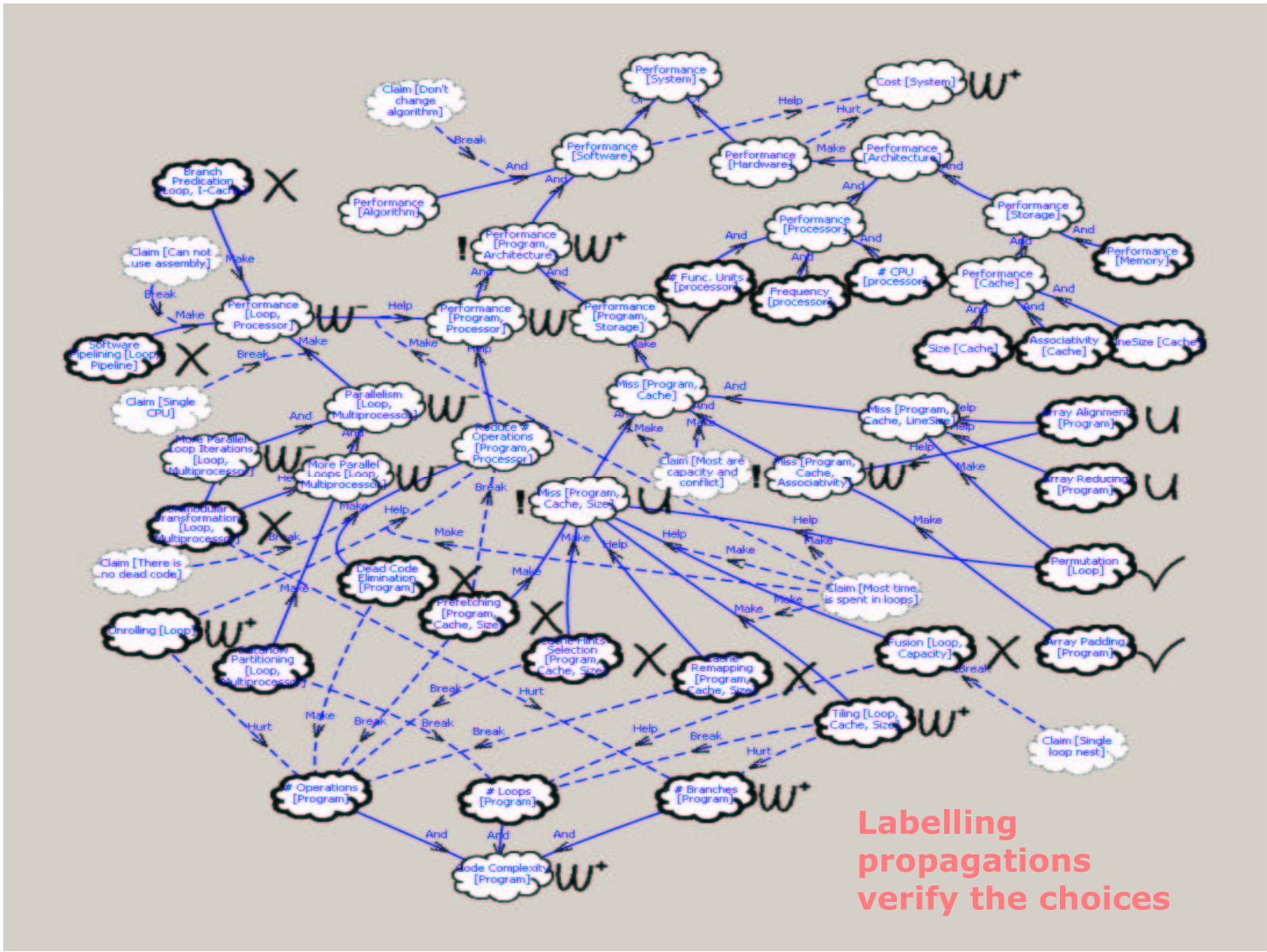


Decomposition of the performance soft-goal


Performance [System]
Cost [System]
Claim [Don't change algorithm]
Performance [Software]
Performance [Hardware]
Claim [Not much instructions]
Branch Predication Loop, I-Cach
Performance [Algorithm]
Performance [Program, Architecture]
Claim [Can not use assembly]
Performance [Loop, Processor]
Performance [Program, Processor]
Performance [Program, Storage]
Software Pipelining [Loop, Pipeline]
Parallelism [Loop, Multiprocessor]
Claim [Single CPU]
Miss [Program, Cache]
Miss [Program, Cache, LineSize]
Array Alignment [Program]
Reduce # Operations [Program, Processor]
Claim [Most are capacity and conflict]
Miss [Program, Cache, Associativity]
Array Reducing [Program]
Miss [Program, Cache, Size]
Permutation [Loop]
Claim [There is no dead code]
Dead Code Elimination [Program]
Claim [Most time is spent in loops]
Prefetching [Program, Cache, Size]
Unrolling [Loop]
Cache Hints Selection [Program, Cache, Size]
Fusion [Loop, Capacity]
Array Padding [Program]
Cache Remapping [Program, Cache, Size]
Tiling [Loop, Cache, Size]
Claim [Single loop nest]
Operations [Program]
Loops [Program]
Branches [Program]
Code Complexity [Program]
QUALITY SOFTGOAL
CLIAM SOFTGOAL
Correlation
interdependency


Performance [System]
Cost [System]
Claim [Don't change algorithm]
Performance [Software]
Performance [Hardware]
Claim [Not much instructions]
Branch Predication Loop, I-Cach
Performance [Algorithm]
Performance [Program, Architecture]
Claim [Can not use assembly]
Performance [Loop, Processor]
Performance [Program, Processor]
Performance [Program, Storage]
Software Pipelining [Loop, Pipeline]
Claim [Single CPU]
Parallelism [Loop, Multiprocessor]
Miss [Program, Cache]
Miss [Program, Cache, LineSize]
Array Alignment [Program]
Reduce # Operations [Program, Processor]
Claim [Most are capacity and conflict]
Miss [Program, Cache, Associativity]
Array Reducing [Program]
Miss [Program, Cache, Size]
Claim [There is no dead code]
Dead Code Elimination [Program]
Permutation [Loop]
Claim [Most time is spent in loops]
Prefetching [Program, Cache, Size]
Unrolling [Loop]
Cache Hints Selection [Program, Cache, Size]
Cache Remapping [Program, Cache, Size]
Fusion [Loop, Capacity]
Array Padding [Program]
Tiling [Loop, Cache, Size]
Claim [Single loop nest]
Operations [Program]
Loops [Program]
Branches [Program]
Code Complexity [Program]
Other quality softgoal favors proritizations
Claim softgoal influences selections
Claim softgoal reflect constraints


Performance [System]
Cost [System]
Claim [Don't change algorithm]
Performance [Software]
Performance [Hardware]
Claim [Not much instructions]
Branch Predication Loop, I-Cach
Performance [Algorithm]
Performance [Program, Architecture]
Claim [Can not use assembly]
Performance [Loop, Processor]
Performance [Program, Processor]
Performance [Program, Storage]
Software Pipelining [Loop, Pipeline]
Parallelism [Loop, Multiprocessor]
Claim [Single CPU]
Miss [Program, Cache]
Miss [Program, Cache, LineSize]
Array Alignment [Program]
Reduce # Operations [Program, Processor]
Claim [Most are capacity and conflict]
Miss [Program, Cache, Associativity]
Array Reducing [Program]
Miss [Program, Cache, Size]
Permutation [Loop]
Claim [There is no dead code]
Dead Code Elimination [Program]
Claim [Most time is spent in loops]
Prefetching [Program, Cache, Size]
Unrolling [Loop]
Cache Hints Selection [Program, Cache, Size]
Fusion [Loop, Capacity]
Array Padding [Program]
Cache Remapping [Program, Cache, Size]
Tiling [Loop, Cache, Size]
Claim [Single loop nest]
Operations [Program]
Loops [Program]
Branches [Program]
Code Complexity [Program]
Considering another quality soft-goal alters prioritizations and selections


Performance [System]
Cost [System]
Claim [Don't change algorithm]
Performance [Software]
Performance [Hardware]
Performance [Architecture]
Performance [Algorithm]
Performance [Program, Architecture]
Performance [Processor]
Performance [Storage]
Performance [Memory]
Branch Predication [Loop, I-Cach
Claim [Can not use assembly]
Func. Units [processors]
Frequency [processors]
CPU [processors]
Performance [Cache]
Performance [Loop, Processor]
Performance [Program, Processor]
Performance [Program, Storage]
Size [Cache]
Associativity [Cache]
LineSize [Cache]
Software Pipelining [Loop, Pipeline]
Claim [Single CPU]
Parallelism [Loop, Multiprocessor]
Miss [Program, Cache]
Miss [Program, Cache, LineSize]
Array Alignment [Program]
Reduce # Operations [Program, Processor]
More Parallel Loop Iterations [Loop, Multiprocessor]
More Parallel Loops [Loop, Multiprocessor]
Claim [Most are capacity and conflict]
Miss [Program, Cache, Associativity]
Array Reducing [Program]
Miss [Program, Cache, Size]
Claim [There is no dead code]
Dead Code Elimination [Program]
Permutation [Loop]
Claim [Most time is spent in loops]
Unrolling [Loop]
Prefetching [Program, Cache, Size]
Fusion [Loop, Capacity]
Array Padding [Program]
Tiling [Loop, Cache, Size]
Claim [Single loop nest]
Operations [Program]
Loops [Program]
Branches [Program]
Code Complexity [Program]
Labelling propagations verify the choices

Measuring Quality Metrics

- Loop unrolling

```
real*8 A(512,512),B(512,512),C(512,512)
do i = 1 , M
  do j = 1, L
    do k = 1, N, 4
      C(i,k) = C(i,k) + A(i,j) * B(j,k)
      C(i,k+1) = C(i,k+1) + A(i,j) * B(j,k+1)
      C(i,k+2) = C(i,k+2) + A(i,j) * B(j,k+2)
      C(i,k+3) = C(i,k+3) + A(i,j) * B(j,k+3)
```

- Loop tiling

```
do i = 1, M, B1
 do j = 1, L, B2
  do k = 1, N, B3
   do ib = i, min(i+B1, M)
    do jb = j, min(j+B2, L)
     do kb = k, min(k+B3, N)
       C(ib,kb) = C(ib,kb)+A(ib,jb)*B(jb,kb)
```

- Loop interchanging

```
real*8 A(512,512),B(512,512),C(512,512)
do k = 1, N
 do j = 1, L
  do i = 1 , M
   C(i,k) = C(i,k) + A(i,j) * B(j,k)
```

- Array padding and interchanging

```
real*8 A(515,515),B(515,515),C(515,515)
do k = 1, N
 do j = 1, L
  do i = 1 , M
   C(i,k) = C(i,k) + A(i,j) * B(j,k)
```

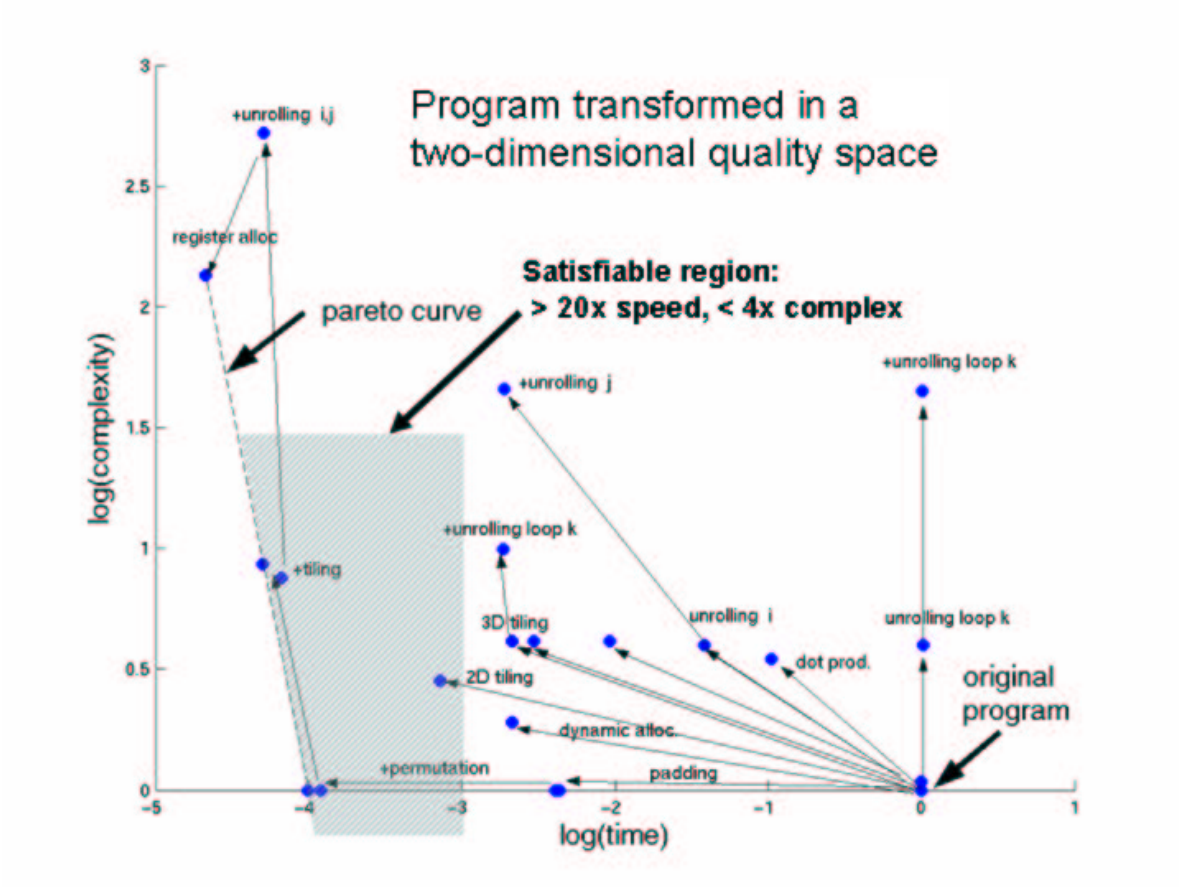
- ... more transformations (equivalent programs)

Measuring Quality Metrics: measured metrics

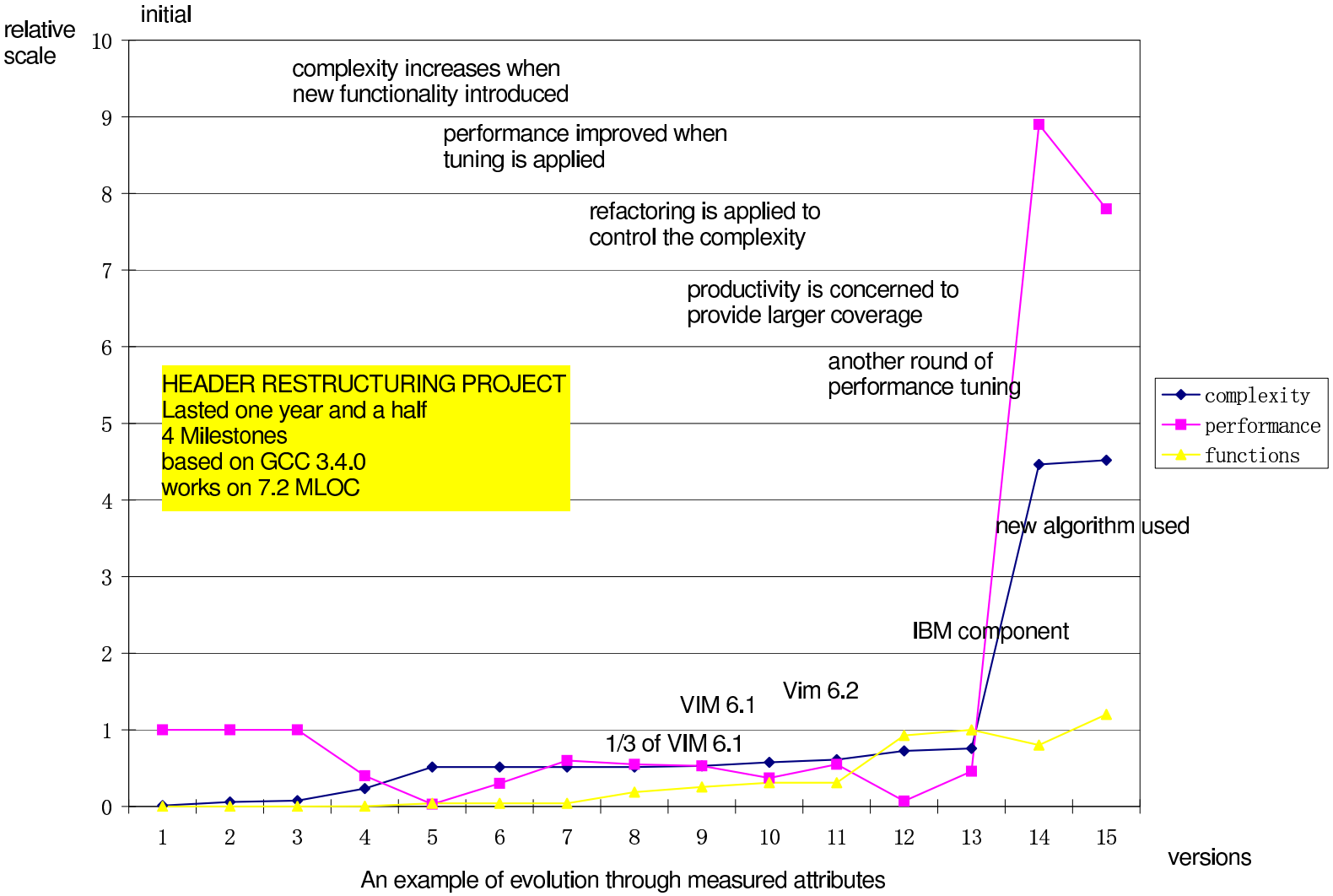
R	time (sec.)	CPI	L1 (10^6)	L2 (10^6)	V (G)	len-gth	vol-ume
1	63.91	64.9	257.9	185.5	4	96	462
2	19.06	20.4	78.6	71.8	4	235	1164
3	4.92	3.36	307.8	1.7	7	185	964
4	1.54	1.33	129.1	47.8	4	96	462
5	5.45	6.30	265.6	12.5	4	96	462
6	1.11	1.23	123.9	44.8	4	96	462
7	3.30	4.28	324.1	2.1	7	312	1682
8	0.89	0.89	81.3	3.0	7	312	1682
...							

Pareto curve and tradeoffs

- ... normalize: $R \to [0, 1]$.



Header restructuring project



An example of evolution through measured attributes

Subjective Quality Factors

- There has been a lot of work on identifying *subjective* factors that contribute to software quality.
- There is a lot of agreement on what factors are important.
- It is hard to define these factors precisely.
- McCall developed one of the more widely used taxonomy of software quality attributes
  - **Quality Factors** - high level quality attributes
  - **Quality Criteria** - lower level attributes used to help measure quality factors.
    [$level \geq 3$? Refer to the Non-Functional Requirements framework by U of Toronto]
- There is remaining difficulty in mapping what we can measure easily (or at all) into what we want to know about software quality.

McCall's Quality Factors[a]

Product Operation	Correctness	Does it do what I want?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Will it run on my hardware as well as it can?
	Integrity	Is it secure?
	Usability	Can I run it?
Product Revision	Maintainability	Can I fix it?
	Testability	Can I test it?
	Flexibility	Can I change it?
Product Transition	Portability	Will I be able to use it on another machine?
	Reusability	Will I be able to reuse some of the software?
	Interoperability	Will I be able to interface to another system?

[a]van Vliet Table 6.5

McCall's Quality Criteria[a]

Access audit

Access control

Accuracy

Communication commonality

Completeness

Communicativeness

Conciseness

Consistency

Data commonality

Error tolerance

Execution efficiency

Expandability

Generality

Hardware independence

Instrumentation

Modularity

Operability

Self-documentation

Simplicity

Software system independence

Storage efficiency

Traceability

Training

[a] van Vliet Table 6.2

ISO 9126 Standard

- Another *product oriented* attempt to define software quality attributes.
  A *user* view of software quality.
- See van Vliet Tables 6.8, 6.3 and 6.4 for a definition of ISO 9126 quality characteristics.
- ISO 9126 doesn’t address software *process* issues.

ISO 9000

- A set of quality standards developed so that *purchasers* of goods can have confidence that *suppliers* of these goods have produced something of acceptable quality.
- ISO 9000 certification has become a widely required international standard. Any supplier who is *not* ISO 9000 certified will find it difficult to sell their goods.
- The ISO 9000-3 standard describes how to apply the general ISO 9000 standard to the software industry.[a]
- The ISO standard addresses design, development, production, installation and maintenance issues.
- The emphasis in the ISO standard is on documentation of the *process* and the managing of the process.

[a] See link on course web page for a translation of ISO 9000-3 into English

ISO 9001 Components[a]

1. Management responsibility
2. Quality system
3. Contract review
4. Design control
5. Document and data control
6. Purchasing
7. Control of customer-supplied product
8. Product identification and traceability
9. Process control
10. Inspection and testing
11. Control of inspection, measuring, test equipment
12. Inspection and test status
13. Control of non-conforming product
14. Corrective and preventive action
15. Handling, storage, packaging, preservation, delivery
16. Control of Quality records
17. Internal Quality Audits
18. Training
19. Servicing
20. Statistical techniques

[a]van Vliet Figure 6.10, also see Appendix A

ISO 9000 Registration

- The effort required to obtain ISO 9000 registration varies directly with how closely an organization's process fits the ISO 9000 model.
- ISO 9000 registration is granted when an accredited inspection organization certifies that the organization's practices conform to the ISO standard. Re-registration is required every 3 years and surveillance audits are performed every 6 months.
- ISO registration can cost a lot of time, effort and money to achieve. It requires continuing effort to stay registered.

A Cynic's View of ISO 9000 Registration

- ISO 9000 Certification focuses on how well the processes are *documented*, **not** on the quality of the process.
- Many companies do the minimum required to achieve ISO 9000 certification for business reasons, but forget about it as soon as the ISO 9000 inspectors have signed off.
- ISO 9000 forces companies to act in ways which make things worse for their customers.
- ISO 9000 is based on the faulty premise that work is best controlled by specifying and controlling procedures.
- Your product and the processes used to produce it can be *absolutely terrible* but you can get ISO 9000 certification as long as the processes are well documented.

The SEI's Capability Maturity Model

The Capability Maturity Model for Software (CMM) is a five level model laying out a generic path to process improvement for a software organization.

1. Initial – ad hoc
2. Repeatable – basic management. processes
3. Defined – management. and engineering processes documented, standardized, integrated, and actually used.
4. Managed – measured and monitored and controlled using measurements.
5. Optimizing – Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

See Watts Humphrey's "Managing the Software Process", Addison-Wesley, Reading, MA, 1989.

CMM Levels and Key Process Areas[a]

1. Initial level
   - No formalized procedures, project plans, cost estimates.
   - Tools not adequately integrated.
   - Many problems overlooked/ignored.
   - Maintenance very difficult
   - Generally *ad-hoc* processes
2. Repeatable level
   - Requirements management
   - Software Project planning
   - Software project tracking and oversight
   - Software subcontract management
   - Software quality assurance
   - Software configuration management

[a] van Vliet Figure 6.13

3. Defined level
    - Organization process focus
    - Organization process definition
    - Training Program
    - Integration software management
    - Software product engineering
    - Intergroup coordination
    - Peer reviews
4. Managed level
    - Quantitative process management
    - Software Quality management
5. Optimizing level
    - Defect prevention
    - Technology change management
    - Process change management

People Capability Maturity Model

1. Initial
2. Repeatable
   - Management takes responsibility for managing its people
   - Staffing, Compensation, Training
   - Performance Management
   - Communication
   - Work environment
3. Defined
   - Competency based workforce practices
   - Participatory culture
   - Competency-based practices
   - Career development
   - Competency development
   - Workforce planning
   - Knowledge & skills analysis

4. Managed
    - Effectiveness measured and managed
    - High-performance teams developed
    - Organizational performance alignment
    - Organizational-competency management
    - Team-based practices
    - Team building
    - Mentoring
5. Optimizing
    - Continuous knowledge and skills improvement
    - Continuous workforce innovation
    - Coaching
    - Personal competency development

Software Quality Assurance (SQA)

- SQA is a collection of activities during software development that focus on *increasing the quality* of the software being produced
- SQA includes
  - Analysis, design, coding and testing methods and tools
  - Formal Technical reviews during software development
  - A multi-tiered testing strategy
  - Control of software documentation and the changes made to it
  - Procedures to ensure compliance with software development standards
  - Software measurement and reporting mechanisms
- SQA is often conducted by an independent group in the organization
  Often this group has the final veto over the release of a software product

SQA Activities

1. Application of Technical Methods
   - Tools to aid in the production of a high quality specification
     i.e. specification checkers and verifiers
   - Tools to aid in the production of high-quality designs
     i.e. design browsers, checkers, cross-references, verifiers
   - Tools to analyze source code for quality
2. Formal Technical Reviews
   Group analysis of a specification or design to discover errors
3. Software Testing
4. Enforcement of standards
   - Specification and design standards
   - Implementation standards, e.g portability
   - Documentation standards
   - Testing standards

5. Control of Change
    - Formal management of changes to the software and documentation
    - Changes *require* formal request to approving authority
    - Approving authority makes decision on which changes get implemented and when
    - Programmers are not permitted to make unapproved changes to the software
    - Opportunity to evaluate the impact and cost of changes before committing resources
    - Evaluate effect of proposed changes on software quality
6. Measurement
    - Ongoing assessment of software quality
    - Track quality changes as system evolves
    - Warn management if software quality appears to be degrading

7. Record Keeping and Reporting

- Collect output and reports of SQA activities
- Disseminate reports to software managers
- Maintain archive of SQA reports
- Maintain log of software development activity (especially testing) to satisfy legal requirements
- Maintain institutional memory of the software development effort

Reading Assignment

van Vliet Chapter 13