Software systems as complex networks
Christopher R. Myers

John McIver

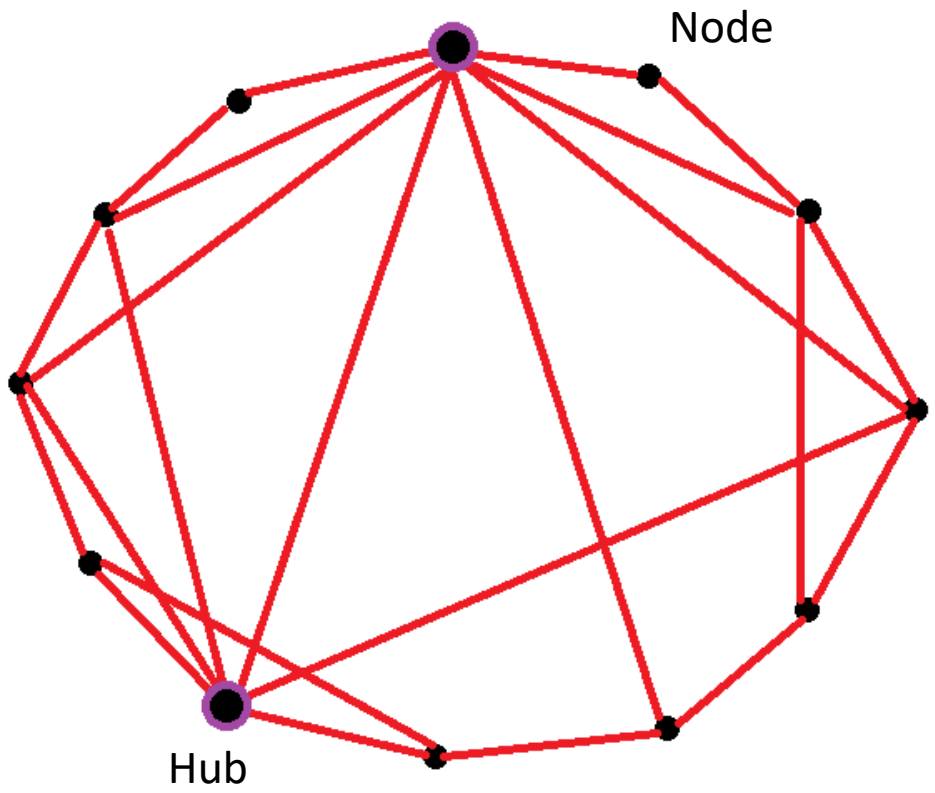
Thomas Liu

Overview of Topic

- Introduction
  - Collaboration in software systems
  - Collaboration graphs
- Results
  - Connected components
  - Degree distributions
  - Degree correlations
  - Clustering and hierarchical organization
  - Topology, complexity and evolution
- Related work
  - Refactoring-based model of software evolution
  - Software systems and complex networks
    - Robustness
    - Degeneracy and redundancy
    - Motifs, patterns, and emergent computational structures

Introduction

- Scale-free networks
  - “a connected graph or network with the property that the number of links originating from a given node exhibits a power law distribution” [1]
  - Internet, World Wide Web, collaborations in science, metabolic pathways
  - Science of complex networks
    - Evolve
    - Robust
    - Adaptive
- Small-world qualities
  - “a network has this property if it has relatively few long-distance connections but has a small average path-length relative to the total number of nodes” [3, p. 238]



[3]

Introduction cont.

- Software systems
  - Interacting units and subsystems
  - Many levels of granularity
  - Organization
    - Highly functional
    - Highly evolvable
  - Evolution though collaborative design
    - Interfacial specificity controlling parameter
    - Evolving biological systems
- Collaboration in software
  - Modularity
    - Reuse
    - Distribution of responsibility
    - Abstractions
  - Optimality of dependencies
- Motivation for study
  - Parallels between software and other complex networks

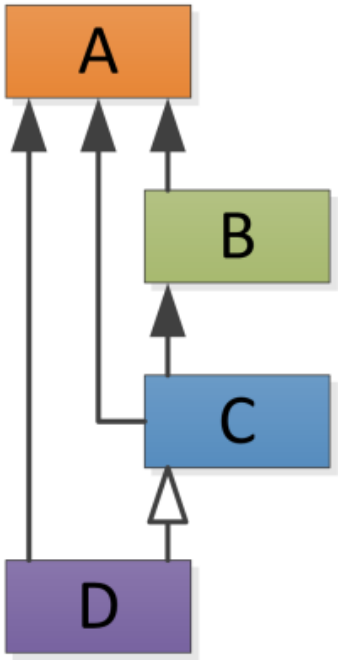
Introduction: Software Collaboration

- Design Patterns for system evolvability
- Mechanism for defining class object interaction
  - Also communicate intent of design
- Micro-structural level
- Solve real issues that can be architecturally cumbersome
  - Double dispatch
  - More examples

Introduction: Software Collaboration

- Call Graphs
  - Static and Dynamic
  - All work in this paper is static based
- Class/Object collaboration diagrams
  - Inheritance
  - Aggregation

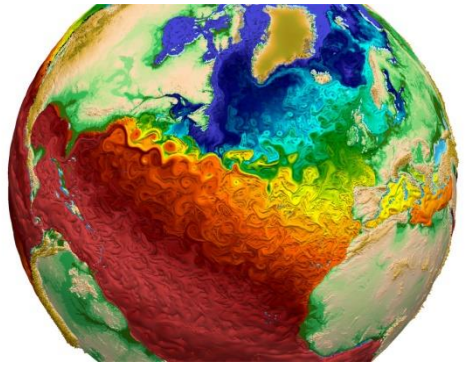
```
#include <memory>
class A {
 // Definition of class A.
};
class B {
 std::sharded_ptr<A> aOjbect;
 // Rest of class B definition.
};
class C {
 std::shared_ptr<A> aObject;
 std::shared_ptr<B> bObject;
 // Rest of class C definition.
};
class D: public C {
 std::shared_ptr<A> aObject;
 // Rest of class D definition.
};
```



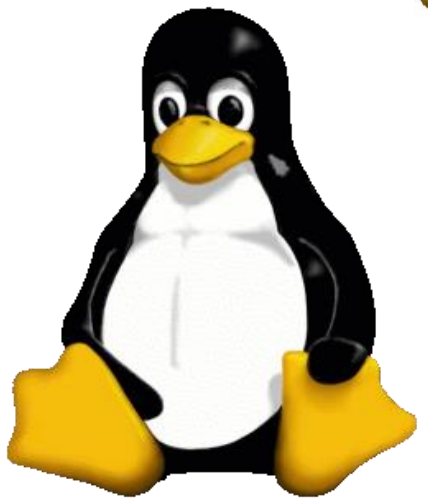
[7]

Introduction: Software Analyzed

- Object Oriented (C++)
  - Visualization Tool Kit (VTK) version 4.0 by Kitware
  - Digital Material (DM) version 1.0.2
  - AbiWord 2.4.19
- Call Graphs (C)
  - Linux 2.4.19
  - MySQL 3.23.32
  - XMMS 1.2.7



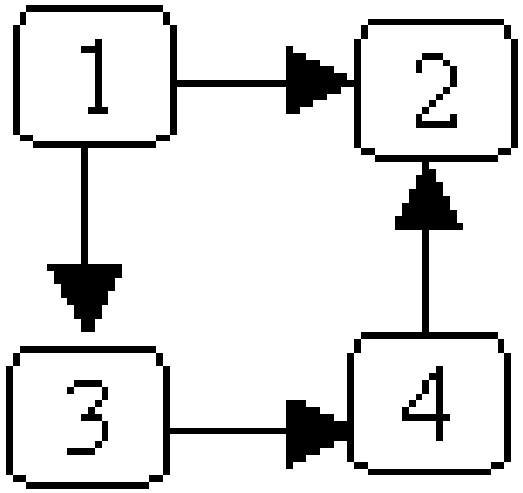
[2]



[6]

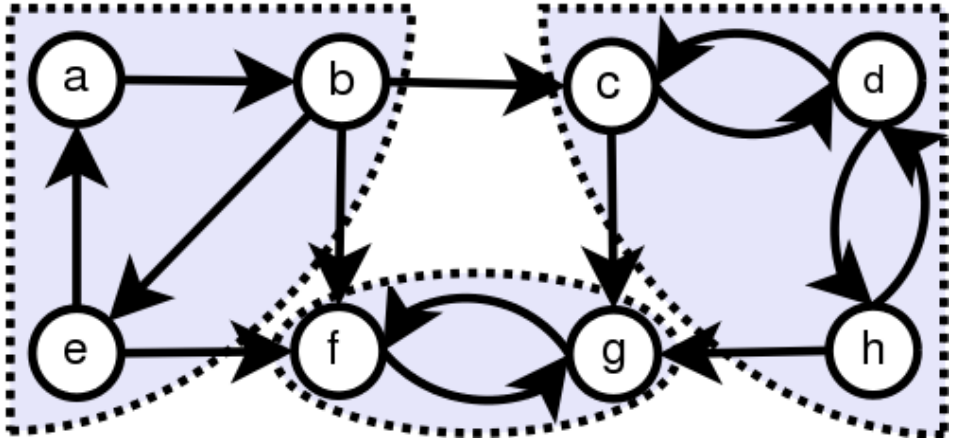
Results: Connected Components

- Weakly Connected Components
  - Found in undirected graph



[5]

- Strongly Connected Components
  - Mutually reachable by traversing directed edges



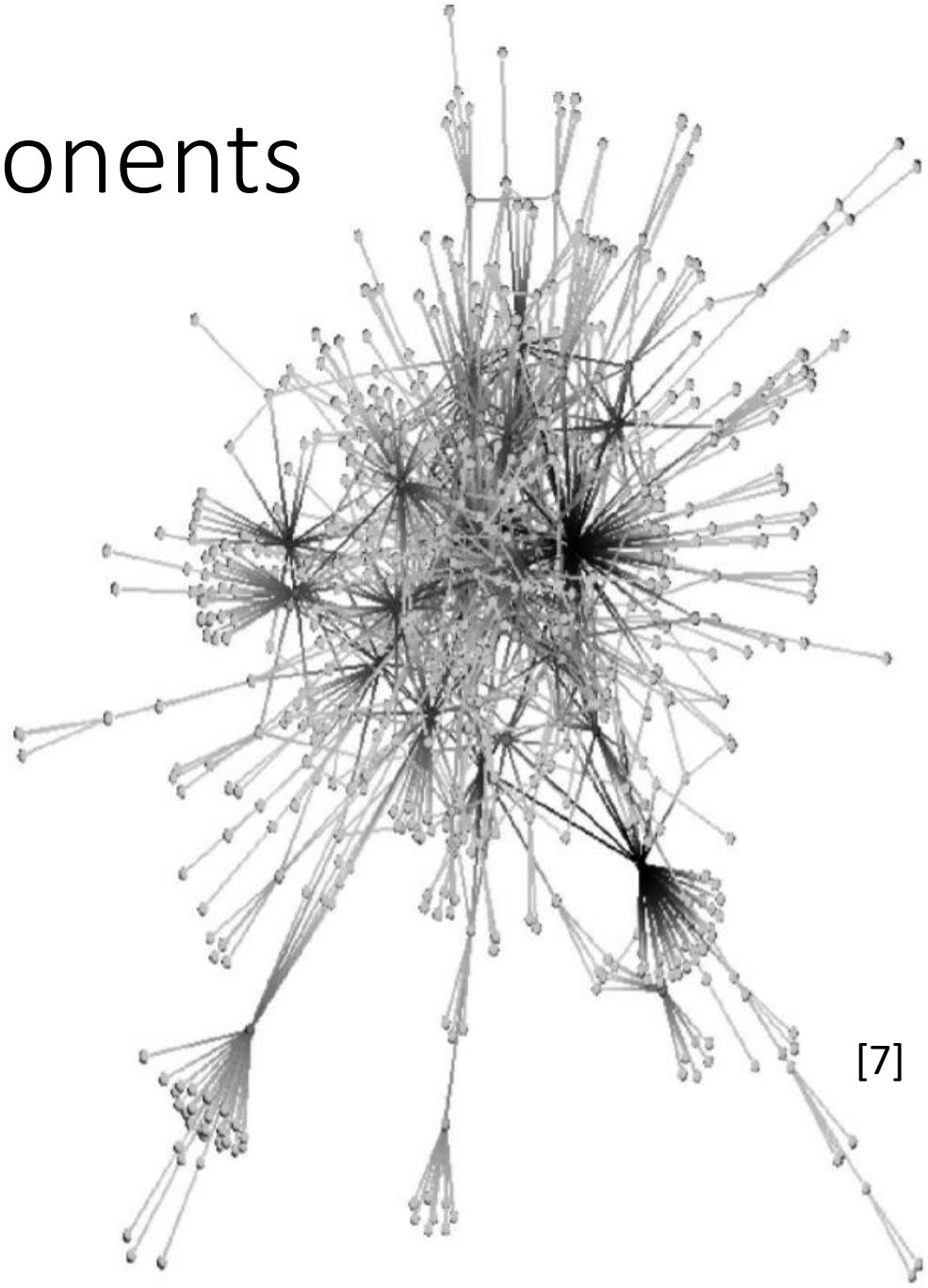
[4]

Results: Connected Components

	VTK	DM	AbiWord	Linux	MySQL	XMMS
# nodes	788	187	1096	5420	1501	1097
# edges	1389	278	1857	11460	4245	1901
# WCC	6	10	19	47	10	36
# nodes in largest WCC	771	162	1035	5285	1480	971
# edges in largest WCC	1374	258	1798	11370	4231	1809
# SCC	4	2	46	10	12	0
# nodes in largest SCC	5	6	25	6	7	0
# edges in largest SCC	8	10	72	9	10	0
Fraction of nodes in any SCC	0.0165	0.0428	0.1332	0.0057	0.02	0.0

Results: Connected Components

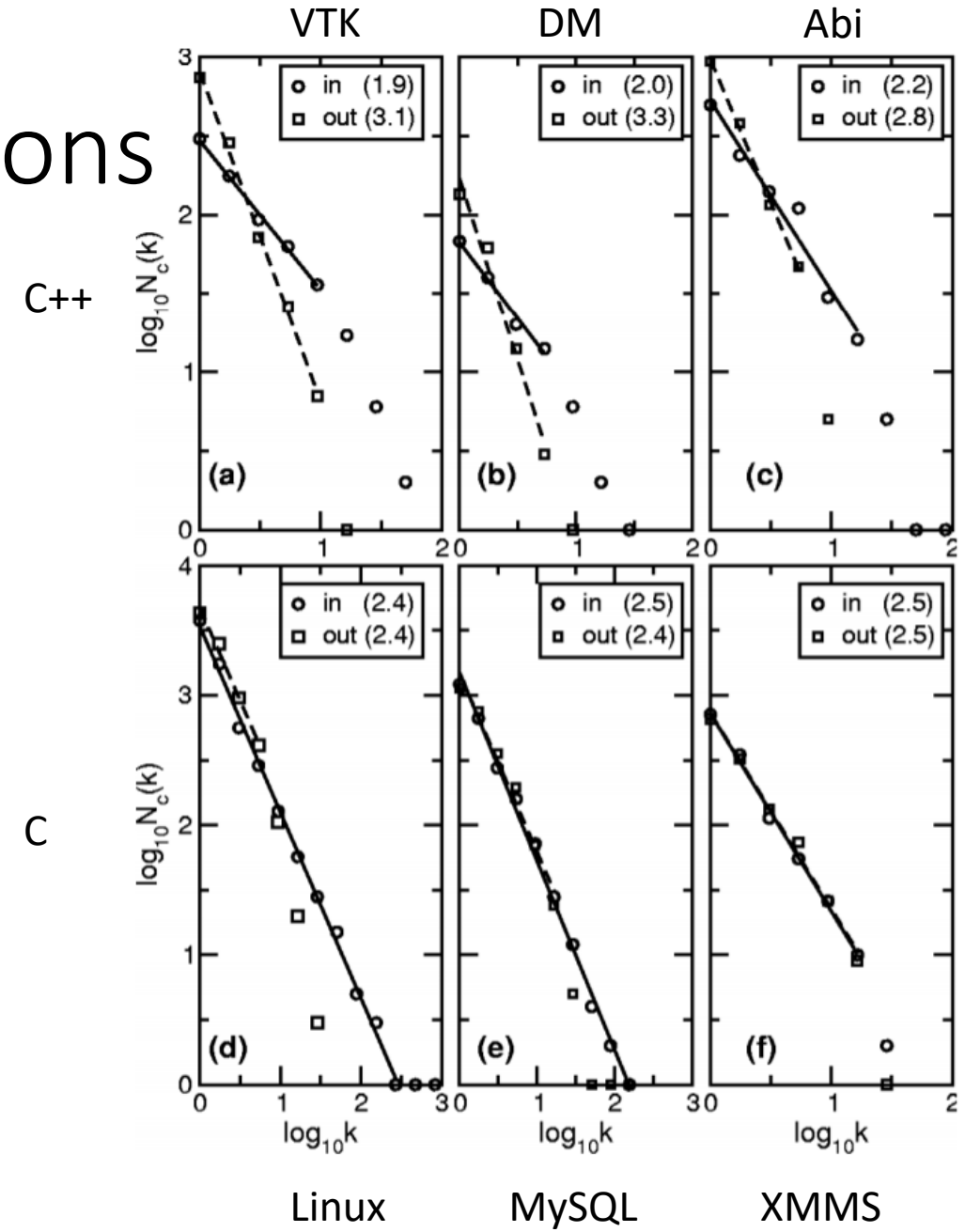
- All six system consist of a single dominant WCC
- Few nodes belong to any SCC
- Lack of strong membership in SCCs
  - different from other directed complex networks
    - WWW
    - metabolic networks
- SCCs reflect subgraphs that are mutually reachable



[7]

Results: Degree Distributions

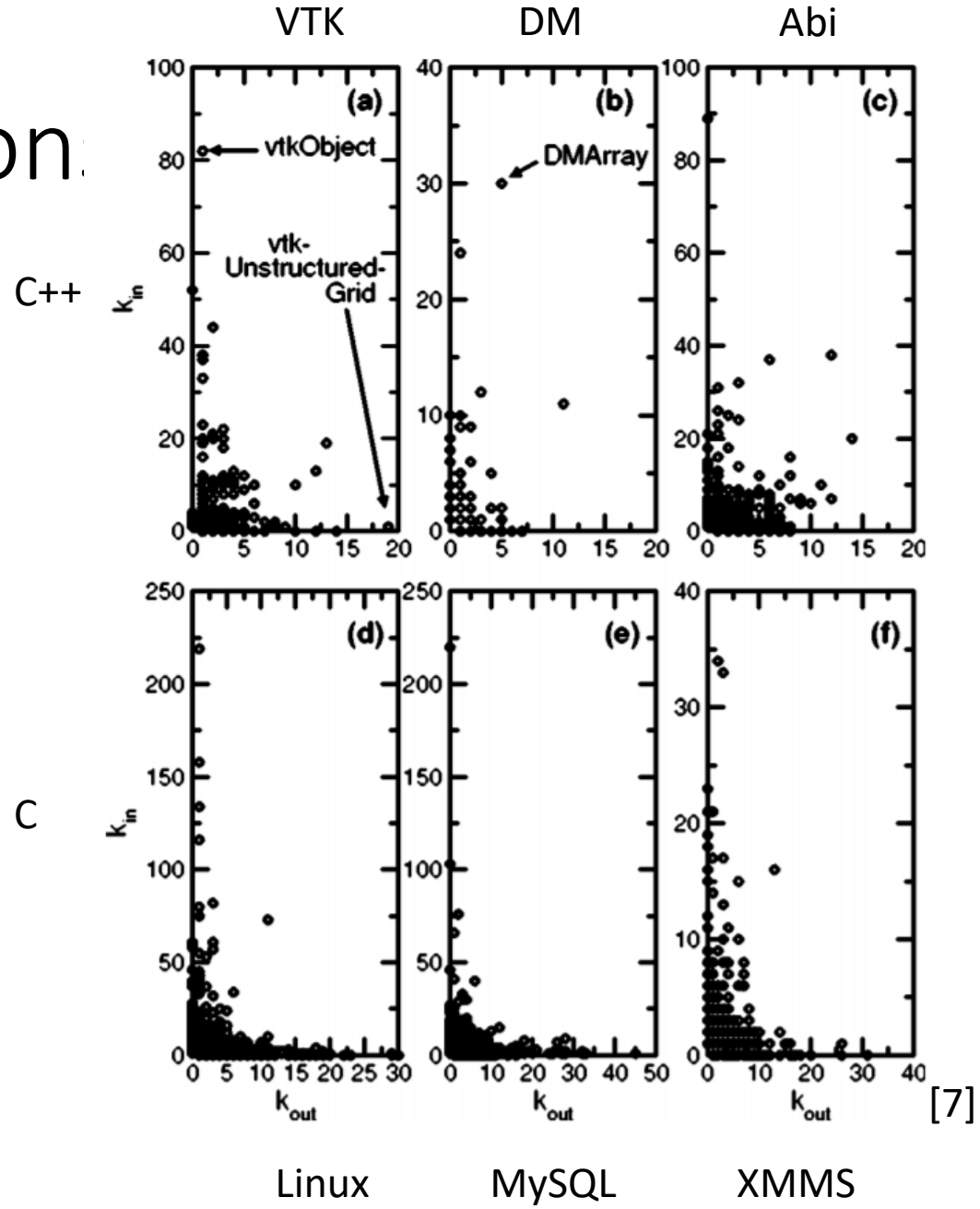
- Summarize connectivity nodes
  - In-degree: k_i^{in}, $P^{in}(k)$
  - Out-degree: k_i^{out}, $P^{out}(k)$
- Scale-free
- Power law over small range
- VTK, DM, AbiWord show asymmetry
  - $\gamma^{in} \approx 2, \gamma^{out} \approx 3$
  - Class collaboration (inherence + aggregation)
- Linux, MySQL, XMMS
  - $\gamma^{in} \approx \gamma^{out} \approx 2.5$
  - Call graph



[7]

Results: Degree Correlation

- ↑out-degree = ↓in-degree
  - vtkUnstructuredGrid aggregates mid-level objects
- ↓out-degree = ↑in-degree
  - Classes like vtkObject are common in polymorphic frameworks

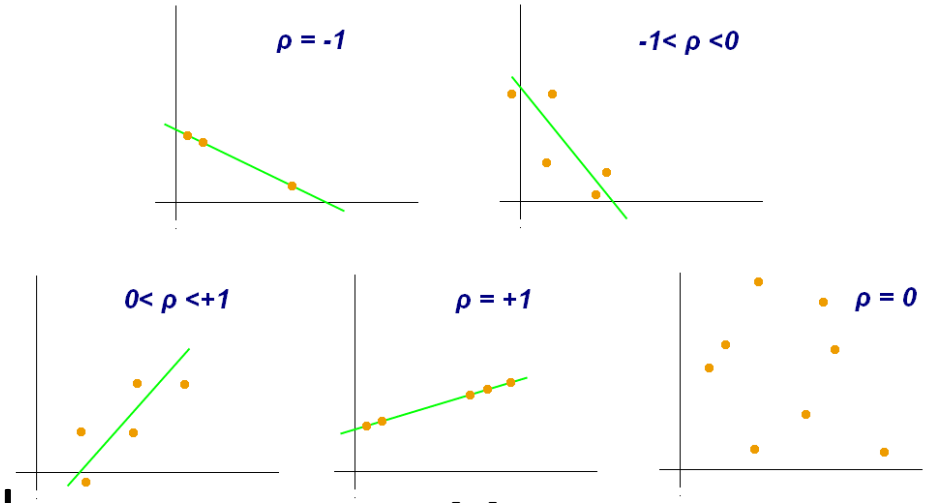


[7]

Results: Degree Correlations

	VTK	DM	Abi	Linux	MySQL	XMMS
k≥10	-0.48	0.01	-0.16	-0.18	-0.23	-0.75
All k	0.09	0.10	0.18	-0.01	-0.03	-0.07

- Pearson correlation coefficients
  - Measures linear correlation
  - [-1, 1]
- k≥10 removes dense core of low degree nodes



[6]

Results: Degree Correlations w/Mixing

	VTK	DM	Abi	Linux	MySQL	XMMS
In-in	0.088	-0.043	0.065	-0.005	0.114	0.067
In-out	-0.034	-0.010	0.083	-0.009	-0.067	-0.036
Out-in	-0.169	0.020	0.042	-0.098	-0.101	-0.180
Out-out	0.137	0.098	0.111	0.014	0.179	0.093
Undirected	-0.194	-0.192	-0.084	-0.067	-0.083	-0.114

- Weak positive correlation among out-degrees means nodes with similar out-degree tend to be connected
- Weak positive correlation among in degrees

Clustering and Hierarchical Organization

- Clustering is typically measured on undirected graphs
- Clustering coefficient $C_i = 2n/k_i(k_i - 1)$ where n is the number of pairs of neighbors of node i that are linked, k_i is the degree of node i
- Clustering of the form $C(k) \sim k^{-1}$ is a signature of hierarchical organization
- Plotting the degree-dependent clustering for the six software graphs does indicate hierarchical organization

Topology, Complexity, and Evolution

- Measured source file size, number of methods, and average revision rate of classes, then compared them to the in-degree and out-degree of those classes
- All three metrics have a strong positive correlation with out-degree, and a weaker, negative correlation with in-degree
- Classes that evolve most quickly tend not to interact directly with each other
- Classes with large out-degrees evolved more rapidly than classes with large in-degrees

A refactoring-based model of software evolution

- Explores how software engineering practices used to enhance system evolvability alter the topological structure of software collaboration graphs
- Refactoring to remove “bad smells” from code that inhibit evolvability
- A simple model involving binary strings of arbitrary length was developed to simulate functions and call graphs.
- After running refactoring processes on the model until the call graph ceased to change for at least 10000 consecutive refactoring steps, many features of observed systems was found in the model

Robustness, fault tolerance, and evolvability

- A by-product of scale-free networks in many systems is enhanced robustness in the face of random node failure
- This is not to be found in software systems, which tend to fail easily when some code becomes mutated
- The complexity in software systems is not for creating fault tolerance but rather evolvability
- Design patterns aim to organize interactions of objects to ensure sufficient specificity for regulation and control without unduly freezing a system into commitments and constraints that are difficult to evolve

Degeneracy and redundancy

- Redundancy is the ability of identical elements to perform identical functions, whereas degeneracy is the ability of different elements to perform similar functions
- Degeneracy plays a role in evolvability but redundancy does not
- Author argues that software collaboration networks do have degeneracy, largely in the form of polymorphism

Motifs, patterns, and emergent computational structures

- Growing interest in scanning large, emergent networks to locate statistical significant, recurring motifs
- Some motifs in information processing systems like gene transcription networks and neuronal systems are feed-forward loops and bifans, as well as biparallel subgraphs
- Examining the six software graphs being studied in the paper, using a motif finding algorithm, the same motifs as above are found to be prevalent
- Extracting complex design patterns from existing software systems without detailed prior information would be useful in identifying functional important motifs

References

1. http://mathworld.wolfram.com/Scale-FreeNetwork.html
2. http://www.lanl.gov/newsroom/picture-of-the-week/pic-week-2.php
3. Complexity a Guided Tour, Melanie Mitchell, Oxford University Press, 2009
4. CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=647584
5. By Kiatdd - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=37108966
6. By Larry Ewing - image sourcedrawing description, Copyrighted free use, https://commons.wikimedia.org/w/index.php?curid=80930
7. Meyers, C., Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs, 2003