A metrics study in Virtual Reality

by

Andrew A. Ray

Keywords: software engineering, metrics, Virtual Reality

Thesis submitted to the faculty of Virginia Polytechnic Institute and State University in partial fulfillment of the requirements of the

MASTER OF SCIENCE

Degree in

Computer Science / Computer Science Applications

APPROVED:

_____

Dr. Sallie M. Henry, Chairperson

_____

Dr. Ronald D. Kriz, Co-Chairperson

_____

Dr. Steven Edwards, Member

May, 2004
Blacksburg, VA

A metrics study in Virtual Reality

by Andrew A. Ray

Abstract

Virtual Reality is a young field and needs more research to mature. In order to help speed the maturity process research was performed to see if knowledge from the domain of software engineering could be applied to the development of Virtual Reality software. Software engineering is a field within computer science that studies how to improve both product and process. One of the sub-fields of software engineering is metrics, which seeks to measure software products and processes. This allows for prediction of certain attributes such as quality. There are several software toolkits that exist in virtual reality that have not had formal software engineering methodologies applied during their development. This research looks at applying knowledge gained from the metrics discipline to the software toolkits used in virtual reality. When metrics are used to measure the toolkits in virtual reality, the metrics seem to behave--produce similar significant correlations--in a similar fashion as when they are applied in previously studied domains.

This work is dedicated to Jesus Christ, my rock and my foundation.

Acknowledgements

I wish to offer sincere appreciation to those who made this effort possible:

My Lord, Jesus Christ, without whom nothing was made that was made.

My parents who prepared me for life after high school.

My family who has shown an interest in making sure I get educated.

My advisor, Dr. Sallie Henry, for guidance, encouragement, and inspiration.

My committee members and other faculty who have helped me through my years at Virginia Tech.

My consultants in the statistics department, Michelle Marini and Ayca Godfrey, without them I would still be reading statistics textbooks.

Dr. Ron Kriz for funding part of my education and giving me the opportunity to work in the VT-CAVE.

Chad Wingrave for proofreading chapter two.

Table of Contents

Chapter 1: Introduction 1
Overview 1
Problem Summary 1
Research Overview 2
Experimental results overview 7
Conclusion and contributions 7
Chapter II: VR 8
Introduction: 8
Input devices: 10
VR hardware: 10
VR Software: 11
VR toolkits: 12
Conclusions: 15
Chapter III: Metrics 17
Introduction 17
Rationale 18
Types of metrics 18
Definition of metrics 19
Chapter IV: Metrics tool and study 23
Metrics collected by the tool 23
Method for calculating procedural metrics 24
Method for calculating structural metrics 25
Method for calculating object oriented metrics: 26
Metrics study 27
Gathering metrics 27
Reorganization of source code 27
Tool use 27
Data analysis 27
Chapter V: Analysis 30
Metrics pairs per system 32
DIVERSE1's unique correlations 33
DIVERSE2's unique correlations 34
DIVERSE3's unique correlations 35
VRJUGGLER1's unique correlations 36
VRJUGGLER2's unique correlations 37
Common correlations for all systems 38
Top twenty metrics per system 42
Comparison to other software domains 44
Metric values between versions 45
Chapter VI: Conclusions and Future Work 50
Discussion of Results 50
Conclusions 51
Contribution 51

Future work 51
Bibliography 54
Appendix: Correlations Between Metrics 57
Vita: 71

List of Multimedia Objects

Figure Title Page

Figure 5.1 Number of statistically significant correlated metrics reported 31
Figure 5.2 DIVERSE1's unique correlations..... 33
Figure 5.3 DIVERSE2's unique correlations..... 34
Figure 5.4 DIVERSE3's unique correlations..... 35
Figure 5.5 VRJUGGLER1's unique correlations 36
Figure 5.6 VRJUGGLER2's unique correlations 37
Figure 5.7 Correlations common among all five releases..... 38
Figure 5.8 Highest correlations..... 42
Figure 5.9 Percentage of highest correlations by package..... 43
Figure 5.10 Comparison to other domains..... 44
Figure 5.11 Object Oriented metric comparison for DIVERSE 45
Figure 5.12 Object Oriented metric comparison for VRJuggler..... 46
Figure 5.13 Structural metric comparison for DIVERSE..... 46
Figure 5.14 Structural metric comparison for VRJuggler 47
Figure 5.15 Procedural metric comparison for DIVERSE 47
Figure 5.16 Procedural metric comparison for VRJuggler..... 48

Tables

Table	Title	Page
Table 5.1	Metric abbreviations……	32

Chapter 1: Introduction

Overview

Computer Science is a broad term that encompasses several different fields. One of these fields is software engineering(SE). SE itself covers many different areas. One such area is measurement. Measurement normally takes the form of quantitatively evaluating a piece of software in order to determine the "quality" of that software. The term "quality", in this instance, implies the ability to maintain and enhance the software. Some measurement techniques predict error-prone software. Through consistent use of measurement techniques in the software life cycle, results can be obtained and interpreted. These results can provide a good measurement of the quality of the software being measured. In order to measure software, certain standards have to be agreed upon. These standards are used as a communication mechanism that interested parties can use. These are called metrics. Metrics cover many different areas of software development. Some are applied to the process of building software, others are applied actual product itself. This research is applying measurement to the actual software product--thereby reaping the inherit benefit of being able to calculate the quality of the software--in the field of Virtual Reality(VR). VR is a relatively new area in Computer Science. VR, like SE, has several sub-fields. The one field of VR that this research is interested in is the VR toolkits being used to make development of applications and content for Virtual Environments easier for the developer/end-user. These toolkits are necessary for VR to grow, and their current maturity level leaves much to be desired. The goal of this research is to investigate applying knowledge from software engineering, specifically metrics, to VR toolkits to see if metrics can be used to improve their quality. Through a more in-depth metrics study of the tools being used, the overall quality of the software involved can be measured, evaluated, and improved.

Problem Summary

VR toolkits are necessary for the young field of VR to improve, and the toolkits current level of maturity leaves much to be desired. In order to help improve VR toolkits this research focuses on applying metrics to VR toolkits. By applying metrics to VR toolkits, observations of how metrics behave with VR toolkits can be gathered. These

observations of behavior can then be compared to behavior observed in other domains. Behavior in this case is defined as producing similar significant correlations between metrics. By comparing the behaviors between other domains and VR toolkits inferences can be made into how to help improve the quality of VR toolkits. The main inference is applying knowledge from metrics to the development of VR toolkits.

Research Overview

This research consists of an analysis of Open Source Virtual Reality(VR) Application Programmers Interfaces (API) with three different types of metrics used to measure the software product. There are two main open source API's being studied in this research. They have been probed and investigated to measure their quality and to allow a comparison between the two of them. Prior to understanding the results of this research, an understanding of the technologies and software being used is necessary. This is why a more in-depth study of VR and the toolkits studied is included in the chapter 2. Chapter 3 contains an explanation of the metrics used in this research. Chapter 4 contains information on the tool used to collect metrics and information about the study. Lastly, the application of metrics to VR toolkits, and the conclusions are presented in the results and conclusions chapters.

Metrics are a way of measuring and analyzing parts of the software life cycle. They can concentrate on the process used to create software or on the actual software being built. Software metrics are the focus of this research. These metrics are standards that measure the actual code that programmers write. They are designed to give an idea of the entire system being built, not to measure individual programmers output. In the case of this research, the types of metrics used were object oriented, structural, and procedural. These metrics address different issues within the software product and each one may not be applicable to every software product. For instance, Object Oriented metrics do not apply to a straight 'C' programs because the 'C' programming language does not have Object Oriented features.

As previously stated metrics, can be applied to different parts of the software life

cycle. The two main types are product and process metrics. Each of these help optimize that specific part of the software development life cycle. This research focuses on the product metrics category since the software has already been developed and released. Process metrics application to VR is important, but is out of the scope of this research. Future work in process metrics is needed. Until one looks at the complete sphere of metrics applied to VR one cannot understand the full picture of how different parts of the VR software product connect.

Measurement of the software product, which is in the scope of this research, is just one reason to use metrics. One of the more powerful side effects of efficient metrics usage is that after a certain amount of time using metrics, problems with the software may be predicted. Software metrics have been around for many years and are an established and accepted way of improving the process of developing software, not to mention improving the actual project. For a real world example of where metrics are applied see the CMM and DOD standard 2167A [PAULKM93] [DOD04]. The benefits that VR can reap from repeated metrics usage are similar to the benefits received with other fields. Probably the biggest contribution to the field of VR software is that if metrics are incorporated into the development process better products will be produced quicker, and will help establish a benchmark platform for the development of other research areas in VR.

VR is a field where the world can be simulated using advanced software and hardware techniques. It is still a young field. The first fully immersive CAVE(tm) was built in 1992, and research is still ongoing to fully leverage the power of this field for academic and industrial use [CRUZC93]. The benefits of virtual training, education, design, investigation, tourism, and entertainment are tremendous. In order to harness the power of VR to support these events, there are many different parts of VR that must be further researched. This research is just one small part of the overall effort in VR research. The software used to develop VR applications today is still very crude [BROOKSFP99]. No Software Engineering methodologies have been applied nor published in the development of VR software. A structured approach to building VR

software should do much to improve the state of VR software.

VR works with a combination of hardware and software. Extensive projection equipment powered by high end computing and graphics hardware provides the visual part of VR[CRUZC93]. In addition to this, specialized input devices and tracking solutions are used to influence the hardware that powers the display system[BROOKSFP93]. The main fields of research in VR are tracking, 3d interaction, display hardware, and graphics [BOWMAN01]. Due to the wide variety of sub-fields converging to create one larger field, VR has had integration problems. In the past, tracking and hardware were limiting factors to VR systems. Now it seems that software is a limiting factor for VR. Regardless of which part of VR seems delinquent research on all the various parts of VR continues.

The software used to create content for VR has been slowly evolving since the early 90's. The first real software API was called CAVELibs [CRUZC93][JOHNSONA98][LEIGHJ97][PARKK00]. It is commercialized and is a proprietary product that has rather expensive licensing costs. As with the current trend of software, when an expensive software solution exists and time permits an open source alternative is created. Open source in this case is GNU GPL/LGPL software. Two different alternatives are currently available to replace the CAVELibs software. These two projects are VR Juggler [BIERBAUMA00][BIERBAUMA01][OLSONE02] and DIVERSE [KELSOJ02][POLYSN04]. Both have been created in academic institutions. They focus on slightly different areas of expertise, but share many common attributes. Due to these similarities, a comparison of the finished products using metrics can be achieved.

The hardware used in the VR environment has historically been monopolized by “big-iron” systems. These are large proprietary systems that provide an all in one computing experience that handles everything from computation and visualization to networking and storage. SGI is the leader in this field and provides the hardware for most CAVE type systems now[BROOKSFP99]. The only problem with “big-iron” is its

cost. The costs hinder the dissemination of VR, and there is a trend to move to more commercial off the shelf systems (COTS). These systems introduce more complexity due to the move away from the integration on one large multi-processor machine [SCHAFFERB04] [OLSONE02].

As stated, VR is a rather complex field. As with any field, it is impossible to look at everything in that field. The natural reaction to this is to divide and conquer and pick specific parts to study. This research does this by focusing on software used to create applications in VR. This area is the foundation area that is the glue for all of the different areas in VR. All of the different areas have to be integrated so that the users and developer do not have to worry about combining and interfacing with all of the different products from VR research. The two projects selected for study cover an important part of the VR market that make it more accessible to ordinary people.

In order to compare these two products, a metrics tool is required. Keeping with the open source nature of this thesis, the tool used is CCCC, which was a PhD dissertation project by Tim Littlefair [LITTLEFAIR01]. A portion of the dissertation involved the creation of a metrics tool, written in C++ , analyzes C++ and Java code for procedural, structural, and object oriented metrics. It produces output in both HTML and XML forms, and is easily transported into SAS for statistical analysis. This is beneficial because the raw results are simple to view and are easily transportable to more powerful tools. The tool is freely available from http://cccc.sourceforge.net. This tool was chosen because it is the only publicly available and free tool found that covers object oriented, structural, and procedural metrics. There are commercial tools available that may be more suitable than this tool, but due to the un-funded academic nature of this research, the only tool usable was this one.

There are five different versions of the software to be analyzed. DIVERSE versions 1.0, 2.3.1 , and 3.0 beta along with VRJuggler 1.0.7 and VRJuggler 2.0 alpha 3. These software packages were developed in parallel and have shared ideas and implementation details. The software packages represent several man years of

development efforts and provide a wealth of source code (several hundred thousand lines worth) to analyze. These projects have been used in industry, academia, and government institutions for the creation of multitudes of VR applications that cover the whole gamut of VR research [KELSOJ02][BIERBAUMA01].

DIVERSE, started at Virginia Tech, and is an ongoing research effort that focuses on providing usable software for the University, government customers, industry, and by individuals. DIVERSE was started in 1996 by Dr. Ron Kriz and was developed by John Kelso and Lance Arsenault for the purpose of developing a craneship simulator. From previous experiences of building VR applications, it was decided that a base foundation would need to be written so that it would not have to be repeated with each and every application developed. This turned out to be the DIVERSE API that currently runs on LINUX/ IRIX(tm) systems. Most VR development at Virginia Tech was done in a LINUX/IRIX environment. DIVERSE is the smaller of the two projects included. It focuses mainly on I/O devices and uses one graphical API for visualization. The DIVERSE API focuses on making life simpler for the developer while providing flexibility to them. While only having one graphical API being supported can be limiting, it does provide a good end user experience [KELSOJ02].

VRJuggler was developed by Dr. Carolina Cruz-Neira et. al. at Iowa State University. This project was also started in 1996. VRJuggler is an API that focuses on giving the user a portable runtime environment for developing their VR applications. It currently runs on all major operating systems available today. VRJuggler is many times larger (lines of code)than DIVERSE and does not focus on I/O devices or graphical API's, though it provides small hooks for their use. It focuses mainly on providing a base system that a programmer can then customize. This allows for large amounts of flexibility but increases the difficulty in which software can be created [BIERBAUMA00][BIERBAUMMA01][OLSENE02].

While there are differences between the goals of the two software products there is no reason why they cannot be compared. They both have common ground in their

internal code and can be used to develop the same types of applications. Both VRJuggler and DIVERSE can be used in the same environments and perform the same tasks. The approach that both API's take to harness VR content is similar. However, the polish that the end user or developer sees is different. The steps used to develop applications are different. This is a similar situation to most software packages. Very few software packages act similarly in the interface department, even if the functionality is the same. For the two packages described above, it is the classic problem of flexibility and complexity applied in a different area [KELSOJ02][BIERBAUMA01].

Experimental results overview

The in-depth analysis of the correlations among the different metrics types are listed in the metrics section of this research. A simple correlation between metrics that are of the same type is performed along with correlating metrics between different types. All of these correlations happen within the same version of the software. An investigation into whether or not the metric response increased between software versions was also performed.

Conclusion and contributions

The main contribution this work provides is that the experimental results suggest that metrics behave the same way in VR as they do in other domains. This means that metrics could be used to help increase the quality of VR toolkits. This knowledge is one of the contributions of this research. The other contribution that this research makes is the problems identified with VR(which are discussed in detail in chapter two), and the proposed solutions (which are also in chapter two). All of these statements will be investigated in the following chapters.

Chapter II: VR

Virtual Reality(VR) is a term that has several meanings. Hollywood has done a great deal to make people think of it as extremely realistic and life like. Movies such as the Matrix and TV shows like Star Trek have made people think that VR is just like real life. This is the meaning of VR to most people. The problem with this view is that VR is not real at the present time, and people are questioning its practicality. Some people want VR to mimic all five senses perfectly, others are happy to be able to have complete artistic freedom about the environment. Current research is ongoing to try to bring Hollywood's dreams to reality, but this is a long way off. At the present time VR is still only being used in production for a handful of application areas. Hardware and software research is ongoing to build newer generations of VR. With each successive generation we get closer to a Hollywood type of view, but we are still quite a ways off.

This chapter reviews VR, and gives a summary of different research areas and an introduction to VR. First, a focus on the physical configurations of the environments is presented. Next, devices in the environment are described. Then, hardware and different varieties of software that are used are described Part of the software described are VR toolkits, which are the focus of this research. Their descriptions are given along with case studies and justification for their existence. Lastly conclusions about VR are listed.

Introduction:

To really experience VR you have to be in a virtual environment(VE). A VE is what provides VR. When you are inside of a VE the amount in which you are engaged with it is called presence. Simply put, presence is how much a user feels that they are inside of an environment. If a person feels that they are a part of a virtual world, then they are more likely to believe that it is real. The current display technologies available in VR are mainly desktops, head mounted displays (HMD), and Cave Automated VEs(CAVE). Each of these different types has different levels of presence. Desktops have relatively no presence because they do not envelop the user. The user can see outside the monitor and if they turn their head the Virtual World disappears. HMD’s

completely envelop the user and provide full physical immersion because they block out everything except for the VE. They are not perfect though because they cannot encompass the full field of view (FOV) that the human visual system can handle. CAVE's are like a cube that has images projected into it[CRUZC93]. Most caves are typically four sided (front, left, right, floor) and provide an almost fully immersive environment as long as the user does not turn around. A fully six sided CAVE is much more difficult to build but can provide a much better experience because a user is truly inside of the Virtual World. There are one or two walled variations of CAVE type systems that provide immersive desks or workbenches. Two of the problems that are present in all of these different environments that VR is bulky equipment and wires. Both of these can inhibit presence. HMD's are cumbersome and can produce fatigue if they are worn for long periods of time. Wires are necessary for tracking, which is a sub-area in VR. They get in the users way, and can occlude the virtual world [BOWMAN01] [BROOKSFP99] [CARLSSONC93].

Tracking the user is an important part of VR because it adds to the presence of the virtual world being viewed by a user. In real life if a person were standing and wanted to look around something they would take a step to the left or the right and look around the object, or they would lean around the object. In order for this to take place in VR, a user must be tracked so that the world can be updated to reflect when the user moves their head, arm, or other tracked body part. This is a powerful aid to making the user think that the world is real. In order for this to work, the tracker system must be able to push the tracker information quickly and have low latency. Most economical tracking systems use wires plugged into a device on the user to keep track of where the user is at in the world. This also provides the ability to have lower latency processing due to a direct link. Some wireless solutions exist, however they suffer from being cumbersome and have problems with latency and cost. Research is ongoing toward making tracking systems more precise and reducing the size / weight of current devices. The commercial devices available on the market are using magnetic, optical, and even acoustic tracking. The more successful approaches being hybrid devise such as the IS900 that uses accelerometers which are periodically updated by acoustic signals. These different types

of approaches have advantages and disadvantages. Some are lower latency than others, some are larger than others, some cost more than others, etc… There is no perfect solution for tracking, it is still an application-specific decision. Researchers must weigh the pros and cons of the tracker against project requirements. Information in this paragraph was gathered from [BOWMAN01][BROOKSPF99].

Input devices:

VR is not to the point where a user can use their own two hands to interact with the world like humans do in real life. Specialized input devices such as pinch gloves, wands, and styluses are used to interact with a VE and are usually tracked. For example, a user can point a wand up in the air and you can have the user fly where the wand is pointing. There are a wide variety of devices that have been and will be designed to help take advantage of the 3D nature of a virtual world. The different types of interactions that they provide are normally limited to actions such as moving a joystick, pressing a button, or pinching with a glove. More exotic methods exist, but they aren't normally available in most CAVE environments. There are three main types of input devices in a virtual world. They are purely active, purely passive, and hybrid. A purely active system is a system that requires the user to physically interact with the device for input to be sent to the system. Purely passive devices give data to a computer regardless of user interaction with the device. A hybrid device combines both active and passive devices [BOWMAN01].

VR hardware:

One of the last components to describe in a VR system is the computer powering the display. In order to manage the different displays to multiple walls of a CAVE, large expensive proprietary systems have been developed by companies such as SGI. These large "big-iron" systems enable high-end stereo graphics, simple synchronization of multiple displays, higher system bus speed, and the use of multiple processors for computationally expensive tasks. The downside is cost. These machines cost hundreds of thousands of dollars and limit the adoption of VR technology by the masses. However, they aid the development of VR because these systems are all in one packages and alleviate the problems of coordinating multiple machines [OLSENE02]

[SCHAFFERB03] [SCHAFFERB04].

The commercial gaming market has removed several of the selling points for the expensive hardware. Free open source software such as Fedora, Mandrake, and Gentoo Linux has enabled a stable and usable Linux system that provides an alternate to the expensive commercial UNIX operating systems that come on high-end machines. Nvidia and ATI have both been pushing the envelopes for low-end graphics hardware for the past several years. Features such as hardware genlock and swaplock are now available for mid-range prices on non-commodity cards. Commodity cards that have these features such are not available and probably will not be because of their limited market share. The network equipment to share data between machines has also increased in speed and decreased in price and latency. Gigabit networking is sufficient for most applications and is within small budgets [OLSONE02] [SCHAFFERB03] [[SCHAFFERB04].

To recap, the advantages of an SGI system are smaller setup time, simple one machine mental model, integration, no real memory limits, and no synchronization problems. It is a simple cost/benefit problem. For most research institutions cost is something that is prohibiting so clustered VR systems are starting to be used more.

VR Software:

Software integrates displays, graphical toolkits, tracker data, input device information, and all of the other parts of the environment. Several of these tasks have already been researched in other areas of Computer Science.

Data transmission is one example of this. Most of the input devices are hooked to a single machine that interfaces to the specialized hardware. With a clustered system, this is a simple producer-consumer problem. Due to the reuse of knowledge developing systems for general usage has proceeded quickly. For example, the replacement system for the Virginia Tech multi-processor computer was developed and demonstrated to the public in just two months. The performance of the software is not going to compare with the monolithic multi-processor computers, but it does provide a usable alternative to

facilities that do not have hundreds of thousands of dollars to spend on an expensive computer. Most of the software available to do such distributed systems is available for public usage. This also dramatically cuts down on the cost necessary to incorporate one of these systems.

VR toolkits:

The companion to the cluster software is graphical software that allows the end users of VR technology to put their content inside of a VE. There are two levels of software for VEs, graphical software and toolkits [SCHAFFERB04].

Graphical software manages the graphical content. It could be as simple as allowing the use of the industry standard OpenGL language, or it could be as complex as a scenegraph that is built on top of OpenGL. Almost every graphics package has some sort of support for OpenGL because it is the standard low level language of graphics for VR [BROOKSFP94].

On top of this are the toolkits that have been built to provide higher level graphics abstractions. A proprietary example of this would be SGI's OpenGL Performer product. An Open Source example of this would be Open Scene Graph. These graphical toolkits are what most people use to develop content for VEs. In the beginning when VR was just starting to develop graphical packages were the sole focus of development. At this stage graphical packages were tackling model importers, modeling tools, and model formats in addition to rendering. The initial graphical toolkits were not designed for transparent use in different environments with different devices. This is the downside to the early graphical toolkits is that they did not address the devices and displays that are used in VEs. However, this has proven beneficial because of specialization in graphics. Graphical toolkit authors have been able to specialize at what they do best without hampering VR toolkits. This has allowed for more graphics oriented features. The graphical toolkits have maintained a reasonable amount of flexibility and are able to be integrated into VR toolkits with some work [OSLONE02] [SCHAFFERB04].

While graphical package development has not focused on trying to make a VR toolkit, the use of software has solved the problem of integrating components of a VE. Higher level packages that handle the displays, devices, and graphics have evolved to meet the needs of VR. They abstract away differences in hardware and platforms that allow VR designers to focus on content. This is the second level for software in VR. These packages are called VR toolkits and are still relatively new. The wealth of VR hardware has grown considerably since the CAVE technology was invented in1992. With multitudes of different hardware available it becomes extremely difficult to design a graphical toolkit that is aware of the environment where it is running. VR toolkits solve this problem. These toolkits all do the task of allowing users to take the content from a higher level package or even the lower level OpenGL language itself and provide hooks for input devices used in VEs. They also provide the ability to run programs in all of the different environments available without having to write any specific display code into an application [KELSOJ02] [SCHAFFERB03] [SCHAFFERB04] [BIERBAUMA00] [BIEBBAUMA01].

A design pattern can help explain the relationship between these two toolkits. The structural proxy pattern says that it provides a surrogate so that a system can control access to the resource. VR toolkits provide surrogates that interface with desired graphical toolkits. This allows a VR toolkit to provide support for trackers, input devices, displays, and specifically tailor the graphics toolkit to work in any environment. To summarize the idea of VR toolkits you can also use design patterns. A VR toolkit is simply a surrogate that allows access for a user or developer in a VE. A VR toolkit is a façade that contains surrogates for different devices, displays, and graphical toolkits [GAMMAE95].

Two main types of VR toolkits exist, closed and open source, Commercial and community, the struggle between profit and availability. The CaveLIBS is the premiere commercial toolkit, and there are others such as Virtools. The open source VR toolkits first started around 1996 and were available to the public in early 1998. Two early examples are VRJuggler and DIVERSE. Another example is FreeVR. These toolkits

support varied viewpoints of development for a virtual world. Some require users to take a raw runtime engine and build each piece of their application on top, while others allow users to simply insert their content. All are meant to make it easier for the developer and end user to embrace VE technology. In six years much has changed but there is still not a definitive toolkit in this area. Every location seems to have its on special toolkit or uses a modified version of one available. The reason for this is the fact that current focus of everyone in the field is not towards trying to have a unifying toolkit, it is still on the application or on research. People do not want to worry about creating a toolkit when they need to be building application X or interaction technique Y to acquire grant Z. The same is true in commerce because the "killer-app" has not been invented yet. While this attitude of application and interaction technique development persists, and a standard toolkit does not exist, efforts will be fractured, leading to multiple people covering the same ground. Until funding centers push for unification of VR tools or put requirements on integrating work back into a central body of tools this trend will continue. Information stated above from [BOWMAN01] [OLSONE02] [SCHAFFER04] [KELSOJ02].

The need of the VR toolkits has been demonstrated by the equipment available at the University Visualization and Animation Group facility at Virginia Tech. Researchers and developers spend most of their time developing their applications at their desktop. The researchers vary from computer scientists, to engineers, architects, and artists. After they have created an application and are ready to make use of the hardware available they have several options available to them. They have the option of using an immersive desk, an Immersive workbench, a head mounted display, or a CAVE. On top of this haptic feedback, or use of a motion platform in the CAVE exists. Additionally, input devices such as various head trackers, wands, joysticks, usb devices, pinch gloves, and others. All provide different options for the users whenever they are working with their application. Without a VR toolkit, it is next to impossible to incorporate support for all of the different environments and devices available in one application. Many facilities only have one or two of the listed environments and have limited libraries for that piece of equipment. This provides what they need until new hardware is acquired and then

they have to rework all of their applications to work with the new hardware.

In order to manage the different types of hardware available at Virginia Tech, DIVERSE was created. It is an acronym that stands for Device Independent VEs that are Reconfigurable Scaleable and Extendable. It is a two-part toolkit comprised of the Diverse ToolKit (DTK)and the Diverse interface to opengl PerFormer (DPF). DTK concentrates on creating network shared memory, and managing input devices, with several miscellaneous tools to support this. DPF is the Diverse interface to SGI's OpenGL Performer. DPF is built on top of DTK, configures the displays for multiple environments, manages the VR devices, providing easy access to them for the user. This toolkit is approaching five years of age and has been mildly successful with adoption outside of the university with government, military, and private companies. The next generation will add clustering support to the toolkit and will allow for the use of raw OpenGL and all of the toolkits that generate it [KELSOJ02].

A similar toolkit has been developed at Iowa State called VRJuggler [BIERBAUMA01]. The person in charge of this project is also the same person that built the first CAVE and the first VR API, the CAVELibs. FakeSpace and VRCO commercialized both of these original products. This had the effect of slowing the adoption of VR due to the expensive cost associated with acquiring the hardware and software necessary to build a CAVE. Due to this the next generation of the API developed was made open source. From a glance at their project website publication section it focuses on creating the framework for writing VR applications. One strength of VRJuggler over DIVERSE is that they have clustering support available This means that they have developed a solution for making commercial off the shelf (COTS) systems power a VE.

Conclusions:

The problem with VR toolkits is that it seems everyone seems to have their own copy that has been evolved over time. Either the toolkits did not support what was needed or the ever-present problem of not invented here syndrome. The cost of hardware

and software prohibits most facilities from obtaining a wide variety of options. Because of this each of the different toolkits reflect what is available at whichever facility it was developed for. Some toolkits have expanded to support off-site hardware due to collaborative research. This is a step in a positive direction, but still does not solve the entire problem of fragmented toolkit development.

One of the next big steps in VR research will be a push to standardize all of the different toolkits(or have a basic foundation built) so that everyone will have a common platform for doing further VR research. All of the different areas in VR development do not communicate well with the other areas. This causes a large amount of overlap and needs to be taken care of in the future. In order for standardization to happen an exhaustive study of VR will have to take place. An analysis of current graphical and high level toolkits will have to be done. Requirements will have to be generated, and revised multiple times. Decisions on how to best to draw lines around all of the different areas of research will have to be done. After this bridging all of the different areas of research must be made. This research contributes to this process by analyzing two of the different VR toolkits available.

The diversity of VR Tools is a good sign for VR because there are enough different areas being researched and used that research is necessary in order to help integrate all of the different areas. This will be a difficult task because of the extreme flexibility that can be necessary alongside with the ability to abstract away all of the different components so that users will only have to concern themselves with what they want to concentrate on researching / developing /using.

Chapter III: Metrics

How does a carpenter decide if a specific board will fit his job? They measure it. By measuring the board the carpenter can tell specific information about the board. They can tell if it is to long or to short, whether it is to thick or not thick enough. How can a carpenter tell if the board fits these criteria? They use a measurement tool. The information gained from the tool about the board is reported in standard measurement units that every carpenter understands. Not only does the carpenter understand these measurements, but most people in general understand them. For example, inches and meters are rather well known and understood units. The measurements taken can be used during the building process to determine whether or not some item is being developed properly and fits to the plans, or it can be applied after the project is completed to take stock and to validate whether or not the finished product was developed as expected. This is a real world example exemplifies how important and powerful measurements can be.

Introduction

Software measurements are similar to the previous example. They are used both during the software process for measurement and can be used after the fact to analyze the end product. They are collected via a tool and are reported in specific units that are accepted and understood by the community of software developers. People other than software developers can understand the measurements, but they are not as well known as the standard measurements that carpenters use. The way a software measurement tool gathers information differs greatly from the tool the carpenter uses, but the idea is the same. This is also true for the measurements that the software tool takes. Software tools take measurements based on defined and accepted metrics. These metrics measure specific characteristics about software, based on the code that composes the software system it belongs to.

It is worthwhile to note that there are other types of measurement both in carpentry and software. If one analyzes the method with which a carpenter performs his

craft you can begin to form a basis on how to measure effectiveness. This is known as measuring the process that the carpenter follows. There are similar ideas in software development. Processes are the lifeblood of organizations and transcend the products that they build. Measuring processes is much more difficult and is less understood than the measurements of products these processes build. This research does not focus on analyzing the process that develops software, it focuses solely on measuring the product itself.

Rationale

Why would an organization want to measure the software it creates? In order to measure the software it creates, tools must be acquired or built, people trained, results analyzed, and the process of development must be adapted to incorporate all of these changes. In order to answer this question and validate the time necessary to properly implement a measurement program simply look at the benefits that Chidamber and Kemerer listed[CHIDAMBERS91]. Chidamber and Kemerer list that size measures and complexity measures can be used to help plan future projects both in terms of schedule and cost. They can help evaluate the effects of different techniques and tools. They can find trends in productivity, increase software quality, determine future personnel requirements, and deal with maintenance [CHIDAMBERSR91]. These trends are very powerful and can add quite a bit of value to an organization if leveraged properly.

Types of metrics

There were three different types of metrics applied in this research. Each one measures a different type of information and may or may not relate to the other metrics. The three different types of metrics are structural, procedural, and object oriented. Structural metrics cover the interaction among different software modules, not on the modules themselves. Procedural metrics cover the lowest level of the program. They deal with such things as lines of code. Object Oriented metrics take the middle ground of these two groups. They still look at the lower level details of implementation, but they look at the specific building blocks of an object- oriented system. For C++ these metrics are based on classes. These metrics give us a holistic view of the software from the

lowest levels, to the mid range units of the software, to the interactions between these units of software. This allow for the investigation of measurements from different angles [LITTLEFAIR01].

Definition of metrics

The literature divides metrics into various classes; the three classes of interest to this research are: procedural, structural, and object oriented. Procedural metrics, the first metrics defined, were predominately focused on procedural languages such as Pascal, C, Fortran, and Ada. Basically, procedural metrics count tokens in the code.

Structural metrics attempted to measure a different aspect of the software. They view the components procedure or function of a piece of the software and measure the interconnectivity of that component to its environment. The hypothesis of these metrics is that the complexity of a piece of software is influenced more by the communication lines and interconnectivity, than simply by size.

Object oriented metrics were developed in anticipation of measuring the various aspects of a component in the object oriented paradigm. A new paradigm prompted the definition of a new set of metrics.

Procedural Metrics

Lines of code:

The lines of code metric is a size based metric. It is used to communicate the size of the system to others. This metric simply counts the lines of code in a source program. There can be ambiguity issues with the results due to the definition of a line of code [AALBRECHTAJ83] [HALSTEDM77].

Lines of comment:

This is the same as the lines of code metric except it is dealing with lines of comments.

Cyclomatic complexity:

This is a metric developed in 1976 by Thomas McCabe. It seeks to measure the complexity of a procedure or function. The metric is based on the analysis of patterns in a graph constructed from the control flows in a program. This graph must be strongly connected. This allows for analysis of the linearly independent paths of execution within a procedure or function. This metric can be used to compare different programs, and is one of the most accepted metrics. It is designed to be language independent [MCCABETJ76] [MCCABETJ89].

Structural metrics
Fan-in:

Henry and Kafura developed the fan-in metric in 1981 [HENRYS81a]. It is defined as the number of flows of information into a procedure in addition to the global data structures from which a procedure obtains information. For more information on the explanation of flow see the referenced literature [HENRYS81a].

Fan-out:

Similar to fan in, fan out deals with flows going out of the component, and global data structures that the class procedure updates. For more information see on the explanation of flow see the referenced literature [HENRYS81a].

Information flow:

Information flow for a procedure is a product of its fan-in and fan-out metrics. It is a simple formula for multiplying fan in and fan out and squaring it (fan-in * fan-out)^2. The reason for the squaring of fan-in and fan-out was to lend weight to its original definition of lines of code*(fan-in * fan-out)^2 [HENRYS81a]. The lines of code in the original definition was part of the two-fold measure of the complexity of a procedure. The second part of this definition is the connection of the procedure to its environment. This connectivity was weighted (squared) so that its contribution was more than linear. The tool's author gives no explanation on why the length factor was not included. The only inference that can be gathered is that the author of the tool was specifically looking at information flow from a structural metric point of view and wanted to use the

procedural part of the tool to analyze the complexity of the code.

Object Oriented Metrics
Weighted methods per class:

This metric is defined as the sum of the cyclomatic complexities for all of the methods in a class. It focuses mainly on measuring the class itself, not on the interactions of the class [CHIDAMBERSR91].

Depth in tree:

This metric shows the height in an inheritance tree for a specific class. If the class uses multiple inheritance then the maximum length to the top of the tree is used. The farther down a class is in an inheritance tree the more functions that class inherits, and can hinder the ability to predict its behavior. This metric can also show the problems of updating a class higher in the inheritance hierarchy than the class of interest [CHIDAMBERSR91].

Number of children:

The number of children measure simply means the number of children that a specific class has deriving from it. This metric can show the amount of reuse in a system. The higher this metric is, the higher the probability that code reuse is happening [CHIDAMBERSR91].

Coupling between objects:

The coupling between object classes measure looks at how many classes the class of interest is coupled to. This basically means, do the classes interact with each other [CHIDAMBERSR91]. This metric can be used to analyze the maintainability of a class. The higher this number then the more interaction the class of interest has with other classes. This can cause problems when trying to maintain the software system due to all of the interdependencies.

Response for a class:

The response for a class measure deals with how many public methods the class has. It is used to determine the complexity of the object. The larger the number of methods for a given object means a larger complexity for the object. This has ramifications in testing because a larger response from a class requires more understanding of the object from testers [CHIDAMBERSR91].

Lack of cohesion in methods:

The lack of cohesion measure deals with what member variables are shared by member methods. Chidamber defines this metric in two parts. The first part is that all of the member variables a method uses are put into a set. The second part is the number of disjoint sets that are formed by the intersection of the results from the first part of the definition. This means is that this metric looks at which member variables are used in different functions. If member variables are used in several different member functions then the object is cohesive in nature. If member variables are only used in a couple of member functions, then it may be a good idea to split the object into two different objects. This metric is one method to determine if the design of an object is correct [CHIDAMBERSR91].

Chapter IV: Metrics tool and study

Tim Littlefair, in his Ph.D. dissertation, built the tool used to gather metrics for this research. It is written in C++ and is based on a command line interface that uses different front ends for different languages. The tools name is CCCC, and can be obtained at http://cccc.sourceforge.net. The tool uses different output modules and allows for HTML and XML output. The front end (or adapter) for the different programming languages are simply parsers built by the PCCTS toolkit. After the front end runs and parses the code it gives information to be stored in an internal database. This internal database is a relational data model that contains projects, modules, members, user relationships, and extents. A project is one complete software system. For this research each version of the VR software used was a different project. Modules are simply the C++ classes and contain information about member variables, and functions. Members are just one component of a module. User relationships provide the information for inter-module coupling. Extents are simply a term that describes a section of the code where a desired attribute is described. The output module for the software simply goes through the internal database and simply sum together the information in specific ways defined for each metric type. After this the data is written to an HTML or XML file for viewing [LITTLEFAIR01]. This tool was selected because it was the only freely available tool that could be found that covered a broad range of software metrics.

Metrics collected by the tool

There are three types of metrics collected by the tool. They are procedural, structural, and object oriented. There are two types of metrics that can be collected in each of these categories. These two types are primary and secondary. Primary metrics are calculated directly from the source code. Secondary metrics are calculated from primary metrics. For definitions of the metrics provided by the tool see chapter three.

Procedural metrics

LOC: Lines of code

MVG: McCabe's cyclomatic complexity

COM: Lines of comments

L_C: A derived metric that means lines of code per line of comment

M_C: A derived metric that means McCabe's cyclomatic complexity per line of Comment

Method for calculating procedural metrics

The lines of code (LOC) metric is defined by looking at each line as it passes through the lexical analyzer. If the first token of the line is not a comment, not white space, and is not involved with the preprocessor, the LOC counter is incremented. This is not the way that accepted parsers check the lines of code metric, but it is a method that can be used if applied consistently. The normal method is simply counting the number of semi-colons [LITTLEFAIR01].

Lines of comment has the same problems with ambiguity as lines of code. The lines of comments (COM) metric is defined by a counter that is incremented every time a c++ style comment is encountered, at the end of a line inside of a comment block, and when it encounters a */. Also, the tool ignores Rational Rose inserted documentation due to the fact that the author believes it is not truly coming from the programmer and thus should not be counted [LITTLEFAIR01].

The cyclomatic complexity(MVG) measure is incremented each time a token that indicates a decision point is found. For C++, the tokens used are if, for, while, until, ||, && , break, and switch. The reasoning behind this is due to the fact that token counts to approximate the MVG result is a widely used approach. There are problems with this method of calculating MVG but are within an accepted 3-5% error range according to Tim Littlefairs dissertation [LITTLEFAIR01].

Structural metrics

The metrics that tool generates in the structural category are:

FIV: Fan-in that takes into account only publicly viewable functions

FOV: Fan-out that takes into account only publicly viewable functions

IV: Derived metric from FIV AND FOV (FIV*FOV)^2

FIC: Fan-in that only looks at relationships that require the compiler to see the parents code before the clients code can be built

FOC: Fan-out that only looks at relationships that require the compiler to see the parents code before the clients code can be built

IC: Derived metric from FIC and FOC (FIC*FOC)^2

FII – Fan in of an inclusive count

FOI – Fan out of an inclusive count

II: Derived metric from FII and FOI (FII*FOI)^2

Method for calculating structural metrics

These metrics are not as simple to define as the procedural metrics. In order to define these metrics the tools internal information must be accessed and from this relationships are investigated. A relationship can be as a provider of information or a receiver of information. These relationships are generated from inheritance, definitions, and data member declarations, class method declarations and definitions. The information about the relationships is recorded (public / private, etc…). From all of this information the structural metrics can be calculated. The Fan In (FI) metric can be calculated from the receiver relationships. The Fan Out (FO) metric can be calculated from the number of provider relationships. The information flow metric varieties are calculated by their respective fan-in and fan-out counterparts with the formula (FI*FO)^2. The tools author gives no explanation on why the length factor was not included in the information flow calculation. The only inference that can be gathered is that the author of the tool was specifically looking at information flow from a structural metric point of view and wanted to use the procedural part of the tool to analyze the complexity of the code [LITTEFAIR01] [HENRYS81a].

Object Oriented Metrics

WMC1: Weighted methods per class, uses a weight of one for each function

WMCV: Weighted methods per class, uses a 1 for public functions, a 0 for private functions

DIT: Depth of inheritance tree

NOC: Number of children

CBO: Coupling between objects

Method for calculating object oriented metrics:

The five object oriented metrics listed surprisingly were glossed over in the calculation method section of the tool. The explanation was that these metrics were related to parsing of the target language of the dissertation [LITTLEFAIR01]. From analysis of the software internals of the tool, the values are simply counters incremented during the parsing stage of the program. The CBO metric is calculated through sum of the number of provider relationships and the number of receiver relationships. The NOC metric is determined through the relationship map and counting the number of inheritances of derived classes. The DIT metric is determined the same way as NOC, except it is recursively counts where in the inheritance tree the module is until it traces the path to the desired node. The WMC metric and its variants are determined by summing the return values for each method in the class. This will return 0 or 1 depending on the metric definition.

These were the only metrics implemented from the list that Chidamber and Kemerer proposed [CHIDAMBERSR91]. These metrics were the most discussed in literature during the time that the metrics tool was written. Due to structure of tool, it would have to be rewritten from scratch in order to fully implement all of the OO metrics that Chidamber and Kemerer defined. In the future, it will be possible to investigate all of the metrics that were proposed, but in the timeframe of this research it is not feasible. However, this smaller study tells us the initial patterns that appear in the software [LITTLEFAIR01].

All of the derived metrics can easily be calculated given their definition. They consist of ratios and formulas given. These are done after the primary metrics are generated. They are represented in the output just as the primary metrics.

In all, there are nineteen different metrics reported by the tool [LITTLEFAIR01].

Metrics study

Gathering metrics

The packages studied in this research were DIVERSE 1.3, DIVERSE 2.3.1, DIVERSE 3.0 Beta, VRJuggler 1.1 , and VRJuggler 2.0 alpha 3. For future reference these are further referred as DIVERSE1, DIVERSE2, DIVERSE3, VRJUGGLER1, and VRJUGGLER2, respectively. These are different software releases from the respective DIVERSE and VRJuggler development teams. These packages were downloaded from http://vrjuggler.sourceforge.net and from the DIVERSE computer in the UVAG lab. All of the packages were in a standard gnuzip compressed tar format and were extracted by the tar xfz command.

Reorganization of source code

A custom bash script was developed that recursively descended into the directories containing the source code for the DIVERSE and VRJuggler releases listed above. As the script descended it copied all relevant source files (header and implementation files)into a central directory for that package, i.e. copy all headers and implementation files into one directory. In this way, so that the metrics tool could easily process all of the results. The tool itself is not designed to go through different file systems to gather results so the source files had to be relocated in a manner that this tool would accept.

Tool use

After all of the source code was put into a specific directory for that relase, the metrics tool was then pointed at that directory and ran by typing the following command in a UNIX style terminal: ./cccc –lang=c++ /path/to/package. After this, the .cccc directory where the previous command was run (this contained the metrics results) was copied to an appropriately named directory such as DIVERSEX or VRJUGGLERX to denote which package and which release this result belonged to. This directory contained an HTML result file generated by the tool.

Data analysis

From this, Mozilla Firebird was used to view the HTML data format produced from the tool, then the appropriate values for each of the three different metrics sections

were copied and pasted into Open Office and saved as three different text files--one for each section(i.e. PackageName.OO, PackageName.Struct, PackageName.Proc). This process was repeated for each of the five releases studied. After this SAS headers and footers were put into place for each of the text files saved under guidance from the Statistical Consulting Center [SAS01].

After this the Statistical Consulting Center handled the rest of the data processing using two different methods. First, each of the results that the metrics tool produced was put into SAS and analyzed using the CORR procedure.

The format of the data given to SAS was as follows. The metrics tool reported results for all of the metrics on a class basis. There were twenty metrics produced for each class that the tool recognized. The information was in three categories, one for each type of metrics studied. This information was given to SAS and each class was viewed as one observation. From this SAS produced correlations between the metrics by using the observations provided. See appendix A for the results from DIVERSE1, DIVERSE2, DIVERSE3, VRJUGGLER1, and VRJUGGLER2.

A comparison of all of the metrics collected for each specific release was computed. This comparison was a large X by X matrix showing all of the different possible correlations for each release studied. This allowed the investigation of metrics relating to other metrics outside of their group. For example, procedural metrics could be correlated with structural metrics.

From the results generated from the Statistical Consulting Center custom parsing and analysis applications were written to help with the analysis of the data. These results are not shown due to size constraints. For the metric to metric correlations, only those with statistical significance were stored. Next they were parsed and were input into an analysis program that is able to answer different questions such as what correlations are unique to each set of metrics, and what correlations are common to all different versions

of the metrics. The results of this analysis are presented in the analysis section of this research.

Chapter V: Analysis

This section details the analysis from the different tests performed on the metrics. All of the tests that use correlations are based on results from the SAS software [SAS01]. Most of the results in this section deal with correlations between metric pairs. These pairs were recognized from analysis done with SAS. The pairs reported are correlated with a significance level of <=.0. For example, if the metric pair LOC and MVG is in the data and has p value <=.01 then it is included in this count. If the p value was greater than .01 then it would not be included in the results.

This section is broken up into three major parts. The first part is the investigation of metrics that are only found in one of the five versions being studied. The second part is a look at collective results of metrics from the different versions. The third part is an investigation on whether or not the metric results increase from one version level to the next.

Analysis from the initial statistical study on correlations (done using a custom built parser / analyzer)

Figure 5.1 is a bar chart indicating the number of statistically significant correlations from each system.

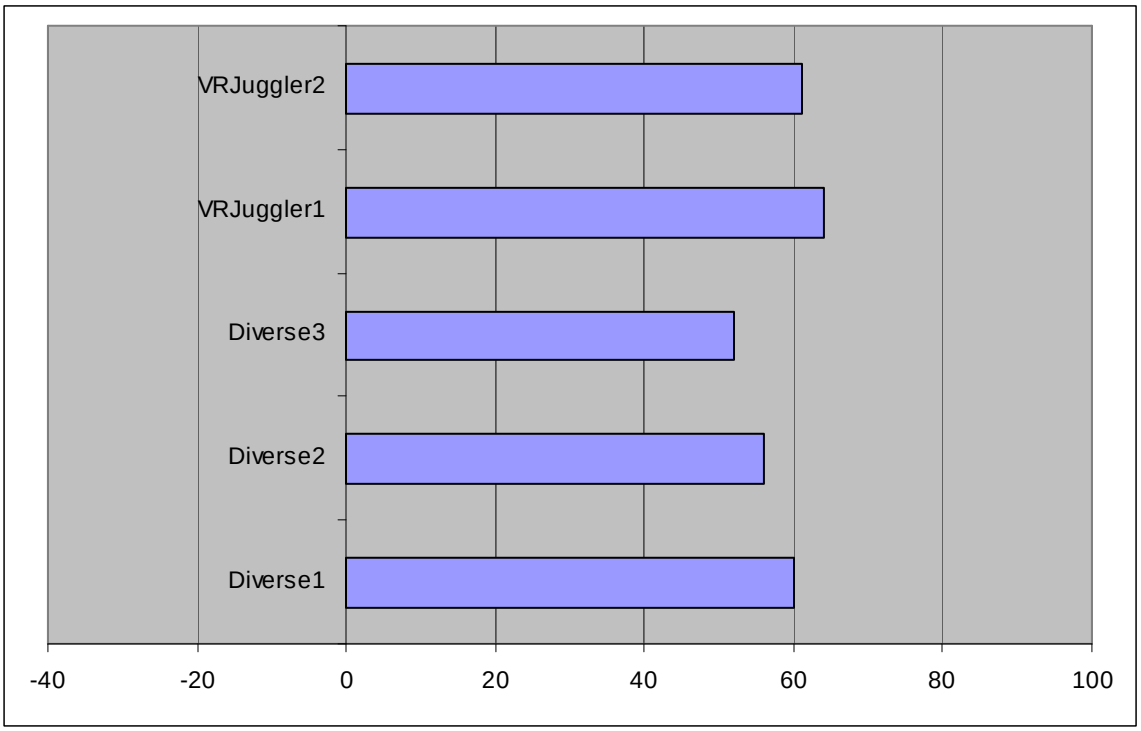

Figure 5.1 Number of statistically significant correlated metrics reported

Figure 5.1 shows each of the systems has a relatively similar number of statistically significant metrics recorded. Each system has a similar numbers of metrics.

Table 5.1 Metric abbreviations

LOC	Lines of code
MVG	McCabe's cyclomatic complexity
COM	Lines of comments
L_C	Lines of code per comment
M_C	Cyclomatic complexity per line of comment
FIV	Fan-in publicly viewable
FOV	Fan-out publicly viewable
IV	(FIV*FOV)^2
FIC	Fan-in Concrete
FOC	Fan-out Concrete
IC	(FIC*FOC)^2
FII	Fan-in inclusive
FOI	Fan-out inclusive
II	(FII*FOI)^2
WMC1	Weighted methods per class
WMCV	Weighted visible methods per class
DIT	Depth In tree
NOC	Number of children
CBO	Coupling between objects

Table 5.1 is a mapping between metric abbreviations and metric names that is provided for ease of comparison.

Metrics pairs per system

This section describes the metrics pairs that are only found in each specific release. This allows observations of the unique correlations in specific systems.

DIVERSE1's unique correlations

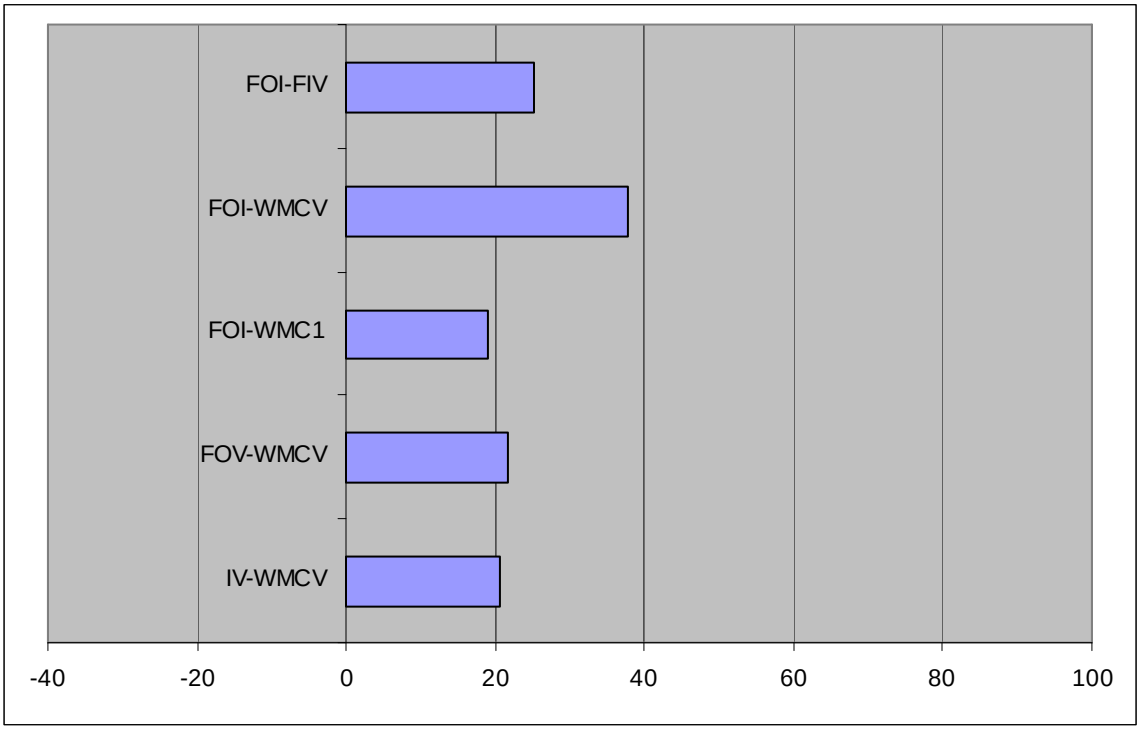

Figure 5.2 DIVERSE1's unique correlations

Figure 5.2 shows the unique correlations for this system are not very high. The three correlations show the derived information flow measure for public functions correlated to the weighted methods per class that are public. It also shows the overall fan out is related to the overall weighted methods per class. The overall fan-out is also related to the fan-in for publicly accessible methods. These results might be explained that in order to have an output to flow to another class you have to have methods in the class for the output. The IV-MCV and FOI-WMC1 are somewhat related because IV is a derived metric that comes from the FOI metric. However, these are such low numbers that there is nothing that can be concluded here.

DIVERSE2's unique correlations

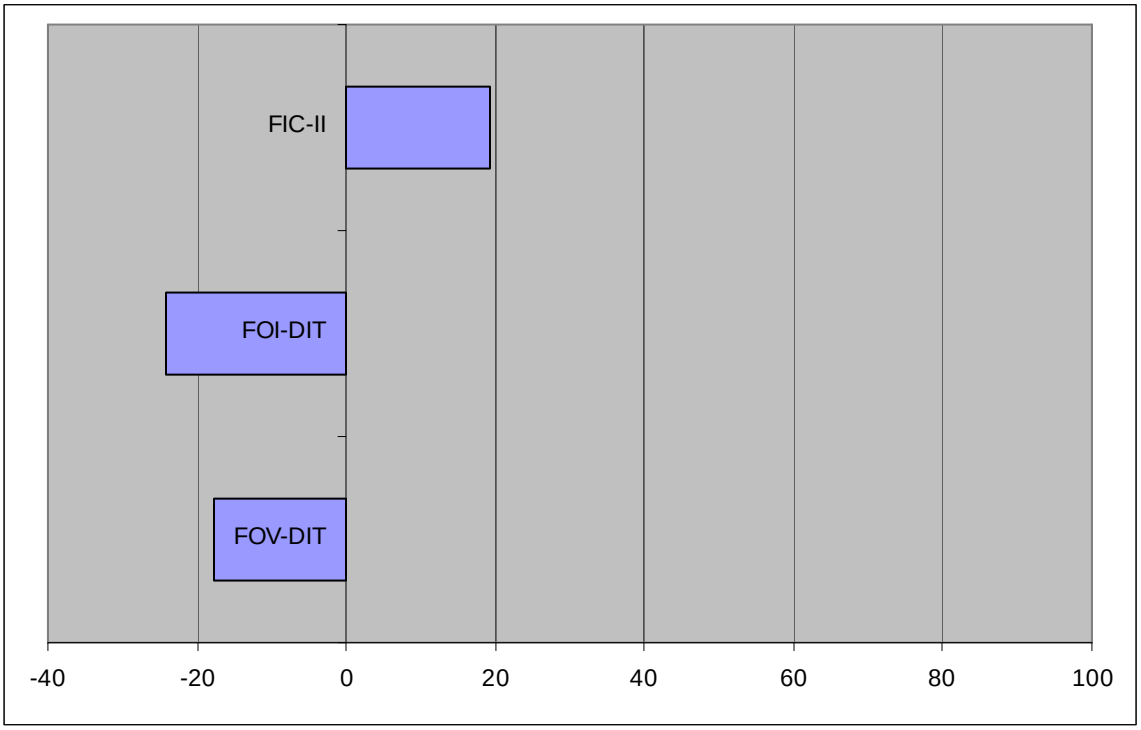

Figure 5.3 DIVERSE2's unique correlations

Figure 5.3 shows another set of low statistically significant correlations. These correlations are somewhat interesting because they do show that fan-in is part of information flow. Since information flow is derived from fan-in this is expected. This is interesting because there are three different varieties of information flow. This is a correlation between two of the different types. The other two correlations show that fan-out is not related to the depth in tree. This makes sense because the depth in an inheritance tree should not influence fan-out in any way.

DIVERSE3's unique correlations

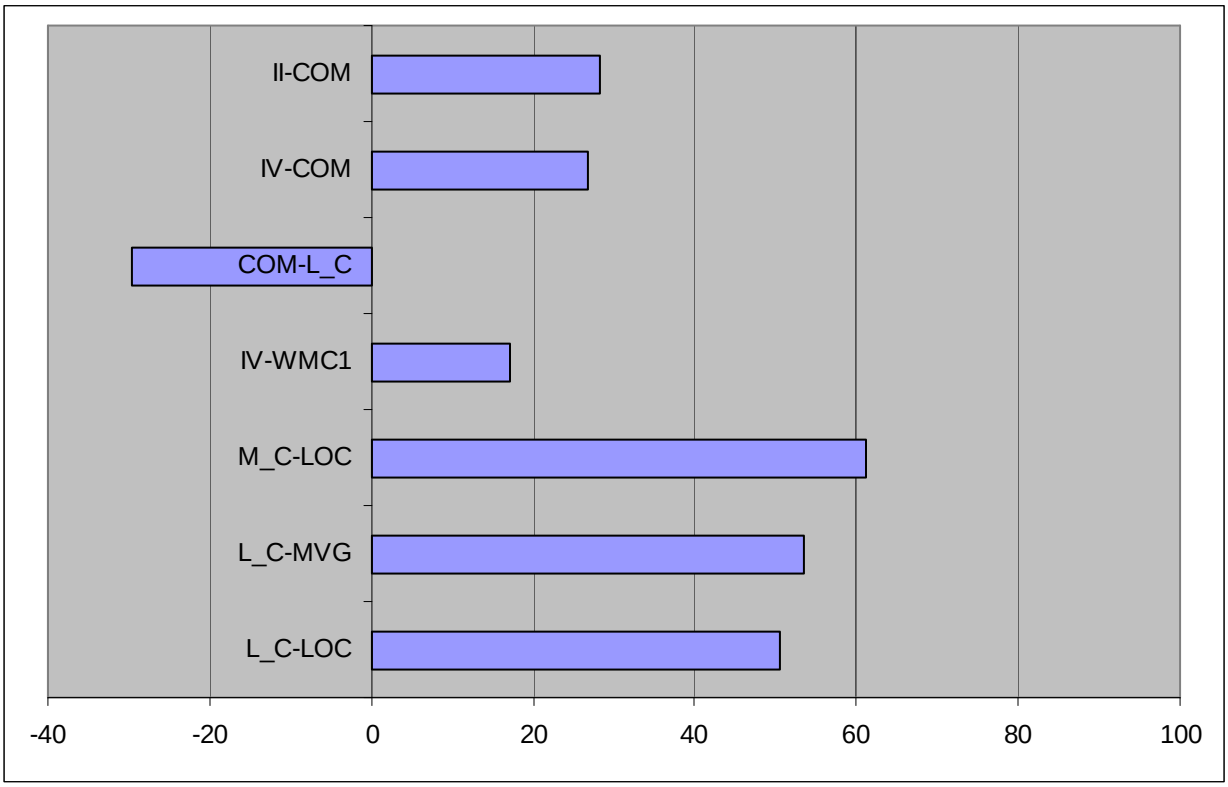

Figure 5.4 DIVERSE3's unique correlations

Figure 5.4 illustrates this system's several unique correlations. These correlations can be explained because of first hand insight into the development process. The first correlation is that the lines of comments is related to the lines of code 50% of the time. In DIVERSE3, an individual was specifically working on documentation for several weeks during development. This produced quite a bit more documentation than normally done on a software system. A similar 53% correlation of the lines of comment and the cyclomatic complexity is because lines of code and cyclomatic complexity is greater than 76% correlated in this version of Diverse. The 76% reference can be found in the appendix. The lines of code per comment compared to lines of code also has a reasonably high correlation(61%). These correlations are unique because one of the goals for Diverse 3 was that a well documented system was produced. The lines of

comments is weakly related to information flow, and is not related to lines code per comments. These are also very weak correlations.

VRJUGGLER1's unique correlations

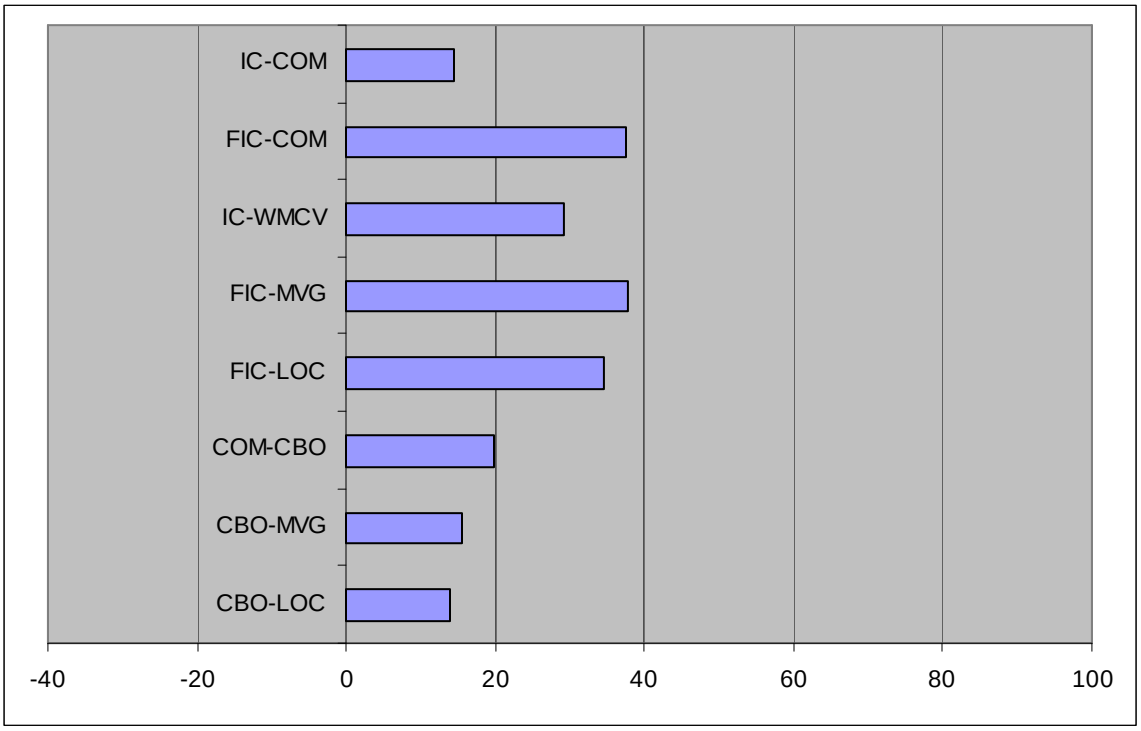

Figure 5.5 VRJUGGLER1's unique correlations

Figure 5.5 shows that coupling between objects was connected to lines of code, cyclomatic complexity, and lines of comment at a low number. Fan-in is related to lines of code and cyclomatic complexity with mid thirty percent. Again, this is a low number. Information flow that requires parent code to be produced is also related to weighted methods per class and lines of comment in a small way. These are interesting trends, but are not significant enough to warrant further investigation.

VRJUGGLER2's unique correlations

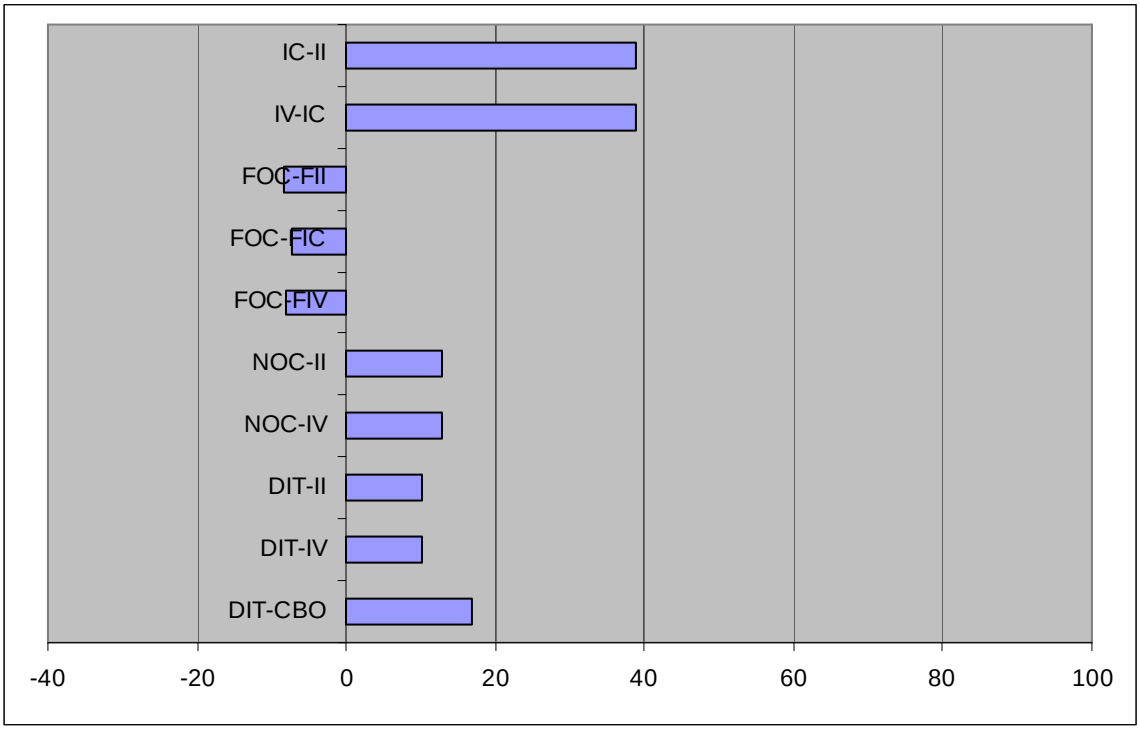

Figure 5.6 VRJUGGLER2's unique correlations

The second version of VR Juggler was easily the largest of the five codebases. By largest, VRJuggler has the most lines of code and number of classes. It may be possible to make the claim that more data means more correlations, as shown by figure 5.6. This codebase also shows that with larger code bases it might be possible that all of the combined information flow metrics are related to each other, as they should be due to their similar definitions. In order to prove this, much larger codebases would need to be used, but this is a rather logical conclusion.

For all of the unique metrics listed in Figure 5.6, none of them were significant at a high level. This means that even though the correlations do show up, it is possible to discount them. The only interesting correlation in this set can be recognized because of the specific decision to invest time into commenting the codebase for DIVERSE3.

Common correlations for all systems

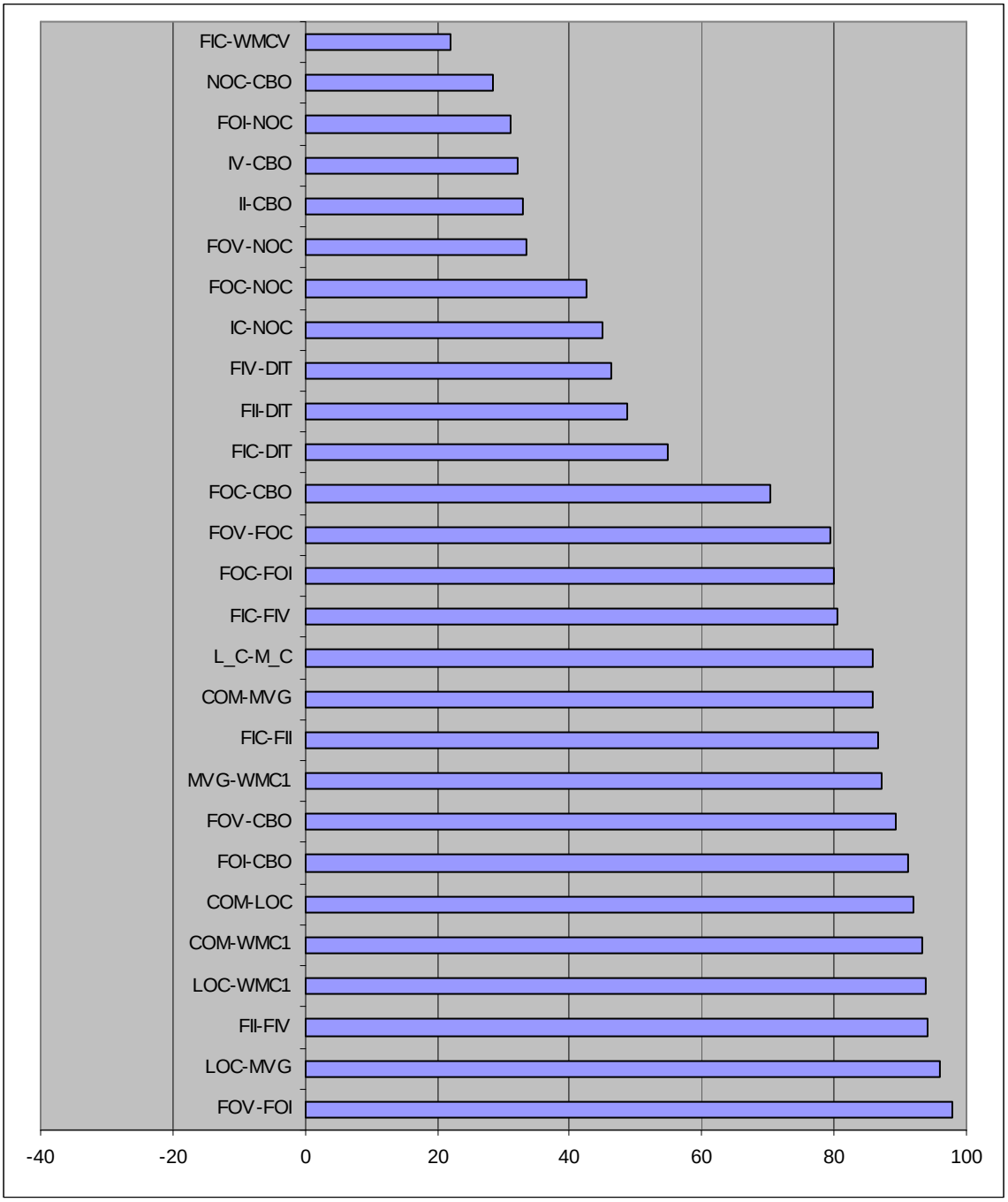

Figure 5.7 Correlations common among all five releases

Figure 5.7 shows the correlations that are common to all five of the systems. There are quite a few normal correlations (from the literature) and some interesting correlations shown from this research [HENRYS81b]. Only the correlations over 80% are

reported. The rest are not accurate enough to warrant description.

First is FIC-FIV. This is a good correlation because it shows that two of the Fan-in metrics relate to each other, as they should due to their definitions. Text book relationships like this should be validated and this is an example of such a validation.

The next correlation is L_C and M_C. This shows that lines of code and cyclomatic complexity are still correlated when the ratio of per line of comment is concerned. This is an interesting result because it shows how strong the link between lines of code and cyclomatic complexity is [HENRYS81b].

The next interesting correlation is COM and MVG correlate together. Not only do lines of code relate to cyclomatic complexity, but so does commenting. This result makes sense considering that most developers have a fixed style of commenting. They either comment a certain amount per project or they comment a certain amount based on the lines of code.

Two versions of fan-out relate to themselves, which is good. This comes from the explanations mentioned earlier in this chapter. Both of them have similar definitions and should be related. This just validates that assumption.

Next MVG relates to WMC1. This shows that cyclomatic complexity is related to the weighted methods per class. This result proves the relationship between the two metrics as given by the definition by Chidamber [CHIDAMBERSR91]. This validation shows that the metrics tool is behaving as expected.

Two fan-out correlations relate to coupling between objects. This could deal with how the tool calculated the metrics or it could deal with an inconsistency of the metric definition. More investigation is required in order to further investigate these metrics. It is more than likely that the tool is at fault for these two metrics being related. If the tool was not at fault then there could be some link between the metrics, which

makes sense considering both them look at information shared with other classes. Both of the metrics are concerned with interconnections between the software, and could possibly be investigated the same way.

COM correlates to lines of code. This is an interesting correlation because it shows that there could be a fixed level of commenting being done based on lines of code or that programmers may only comment at a certain level. It is a good result because it shows that as lines of code increases the amount of commenting also increases.

The next correlations shows COM-WMC1 and LOC-WMC1 are correlated. These correlations might be explained due to the COM-LOC correlation. This would make sense because the lines of code and cyclomatic complexity metrics are highly correlated, and then combining the lines of comment to lines of code correlation would produce this result because the weighted methods per class metric is based off of the cyclomatic complexity metric. Another solution could be that COM and LOC are dependent on the number of methods in order to be calculated, though this should not happen because if it did then the number of functions would be a key measure for software.

After this, another set of fan-in varieties show their inter-relationship. This is again attributed to the definition.

The second highest correlation that is common to all five systems is lines of code and cycolmatic complexity. Dr. Sallie Henry originally demonstrated this result in [HENRYS81b] and it continues to be present in other applications as well [LIW92] [LIW93] [HENRYS84] [HENRYS91]. Code metrics have a history of correlating to each other, after all, they are simply counting tokens in the code. This correlation demonstrates that this trend is true for VR applications as well.

The highest correlation is between two fan-out varieties and shows the strong relationship between definition, assumption, and validation.

Another interesting result that can be found from this analysis is that most of the common correlations are not from the same group of metrics. In other words, procedural metrics and object oriented metrics correlated together. There are a total of sixteen hybrid correlations and whereas there are only 11 correlations that stay within their own category. For example, the procedural lines of code metric corresponds to the object oriented weighted methods per class metrics. This is also true for cyclomatic complexity. Also, lines of comments also correspond to the weighted methods per class method. Also, fan-out correlates to coupling between objects. This last correlation makes sense though because coupling between objects and fan-out are similar in nature due to that they both deal with information going out of the module. It should be noted that these metrics are not the same though. Another interesting result is the number of hybrid correlations between the different groups of metrics. Between procedural and object oriented metrics types, there are two cross pairings. Between structural and object oriented types, there are fourteen cross pairings. Between procedural and structural, there are zero cross pairings. This suggests a relationship between structural and object oriented metrics while also suggesting that there is not a relationship between procedural and structural metrics [HENRYS81b].

Last, these correlations should be considered more valid because they are common among five different observations and most of them have relatively high statistical probability as compared with the metrics that are significantly related and are unique to one system only.

Top twenty metrics per system

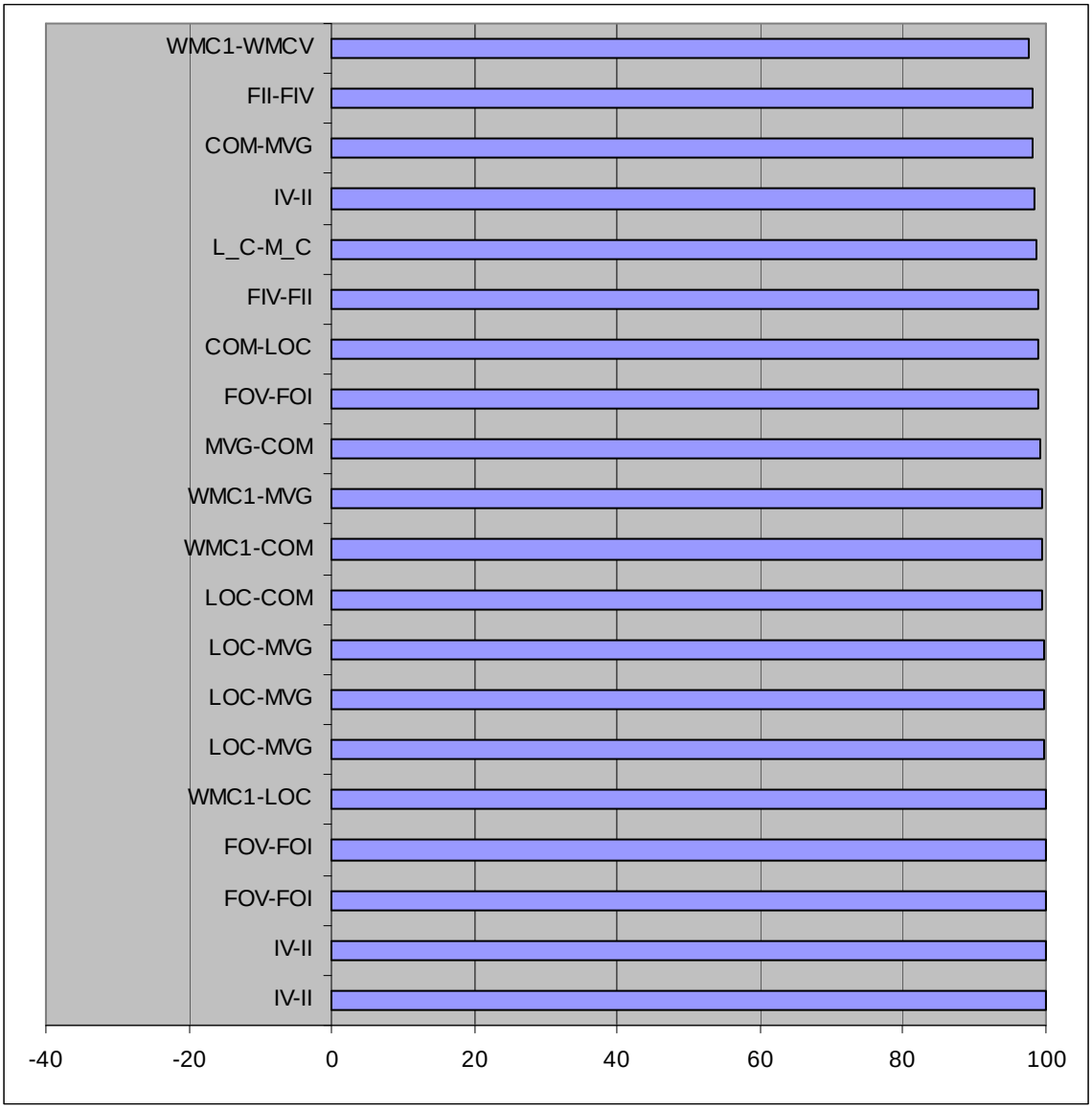

Figure 5.8 Highest correlations

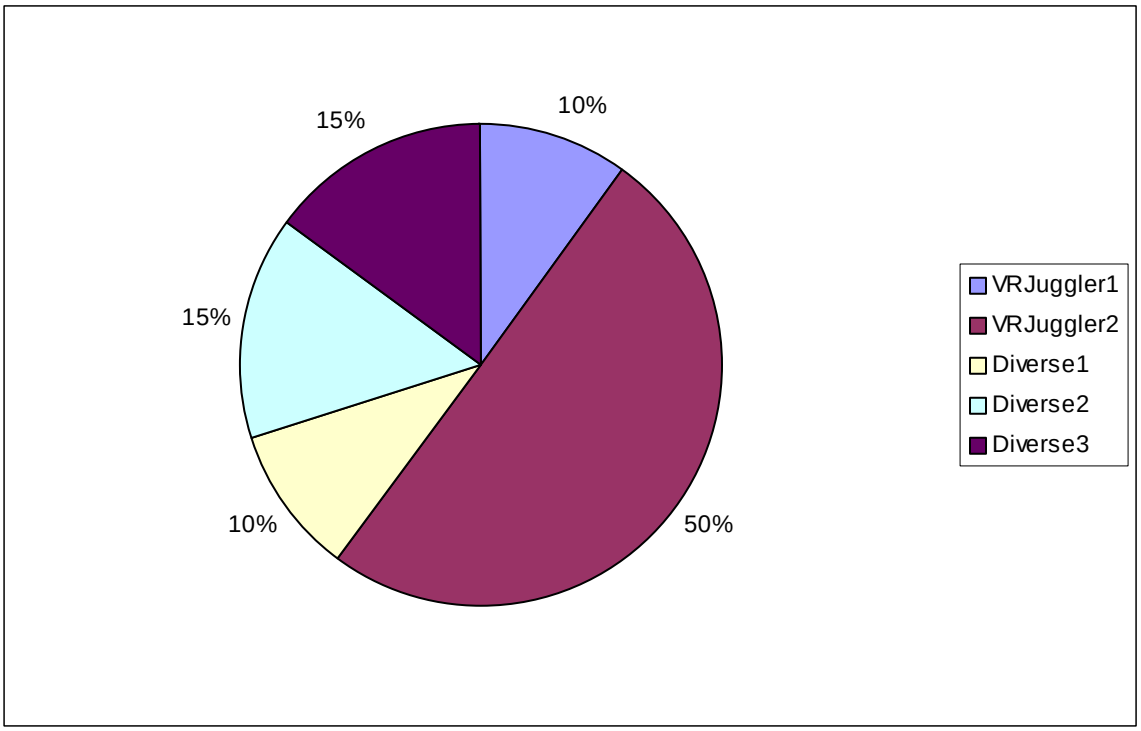

Figure 5.9 Percentage of highest correlations by package

Figures 5.8 and 5.9 represent the highest recorded correlations in the study. In order to compute these graphs, all correlations from the packages studied were put together and sorted based on percentage. Figure 5.9 shows the number of these highest recorded correlations from each release studied. Two major points can be taken from these correlations. First, all of the correlations are highly probable. All of the top twenty are over 97% significant. Second, is that VRJuggler 2 has 50% of the strongest statistical correlations. This could be because that VRJuggler 2 has the highest number of lines of code (159,867). Due to the larger amount of data it may be much easier to determine satisfactory correlations. However, this same argument does not hold true for the next largest codebase because Diverse 1 has the next most lines of code(43021) and only has two correlations in the top twenty where as Diverse 3 has three correlations and has the least number of lines of code(19657). With more data it might be possible to determine if the correct number of lines of code necessary in order to get the most number of statistical correlations. This may not be possible though because a system may be written that is rather boring statistically. This may not hold true for a large enough codebase though. Further research is required.

The rest of the correlations from this list show that almost 50% of the correlations are between variations on metrics types (two versions of fan-out, and so on). The interesting fact is that only the LOC-MVG correlation is repeated more than once(see Figure 5.8). This again shows the strength of the relationship between these metrics again.

Comparison to other software domains

In order to show that the correlations above are similar to correlations gathered from other domains a small comparative study was done. The domains of web browsers and desktop toolkits were investigated. In particular the 3.2.3 release of the KDE project and the 1.8a2 Mozilla releases were studied. The same method used for collecting metrics from the VR toolkits was used on these releases. From the metrics gathered the MVG and LOC metric was chosen due to proliferation of previous literature on it. The following graph shows the results of the study.

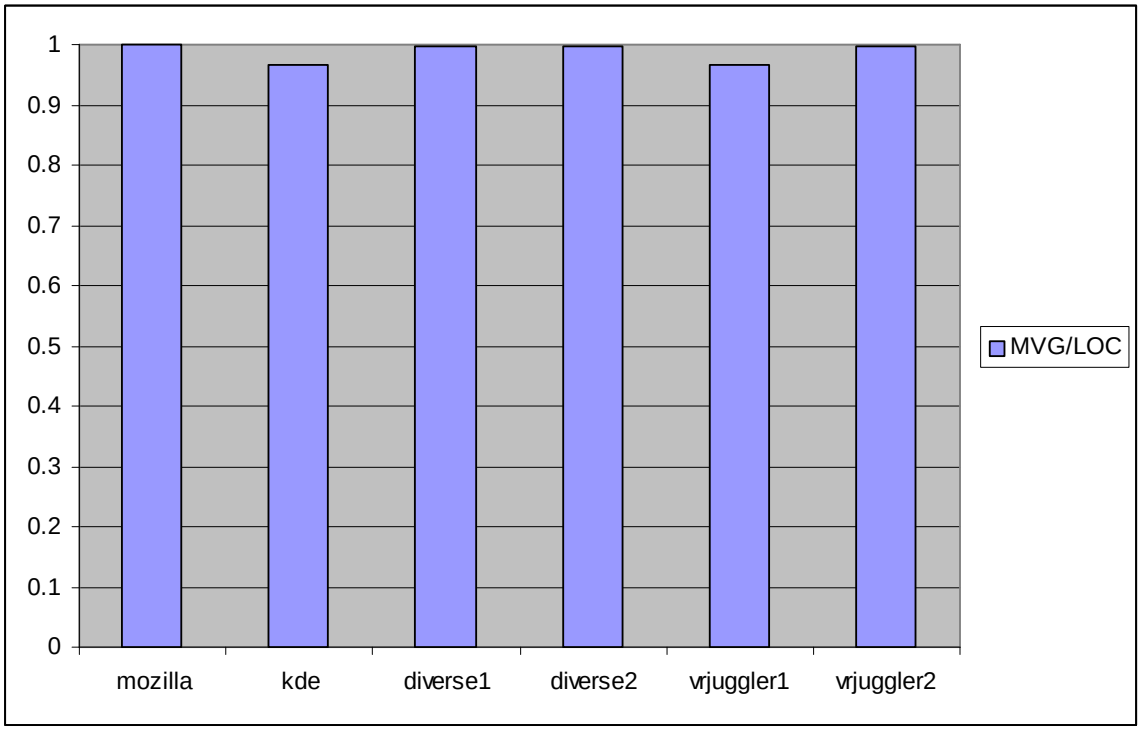

Figure 5.10 Comparison to other domains

Figure 5.10 points out that the correlations between MVG and LOC are similar not only across VR toolkits, but also across the other two domains studied. This helps suggest that the correlation results from VR toolkits are similar to other domains, thus suggesting that metrics might help the development of VR toolkits because metrics have helped improve quality in other domains.

Metric values between versions

The last test done was a comparison between two versions from the same software system. DIVERSE3 was not chosen because it was a revolutionary instead of evolutionary development. This means that it was not an extension, it was a rewrite. This test centered on counting the metrics from the later version of the software that were greater than or equal to the metrics of the first version. This is a very important test and turned out to give us the most positive results. The reason why this test is important is because it shows that software complexity increases over time. The following are lists of graphs for the DIVERSE and VRJUGGLER comparisons for object oriented, structural, and procedural metrics. The lighter bars stand for the number of classes that have greater metric values, and the darker bars stand for classes that a lower metric value.

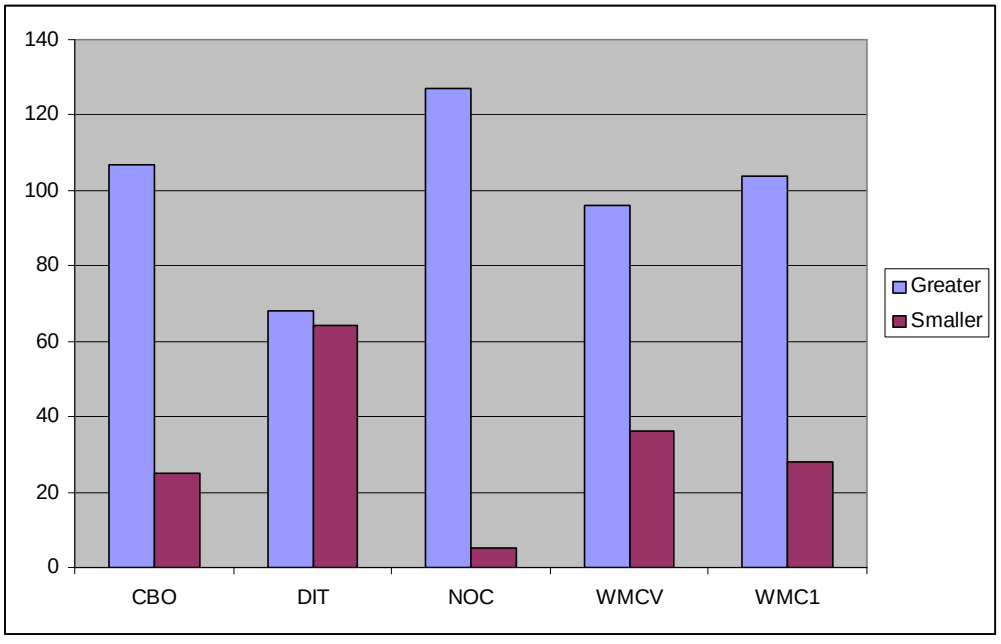

Figure 5.11 Object Oriented metric comparison for DIVERSE

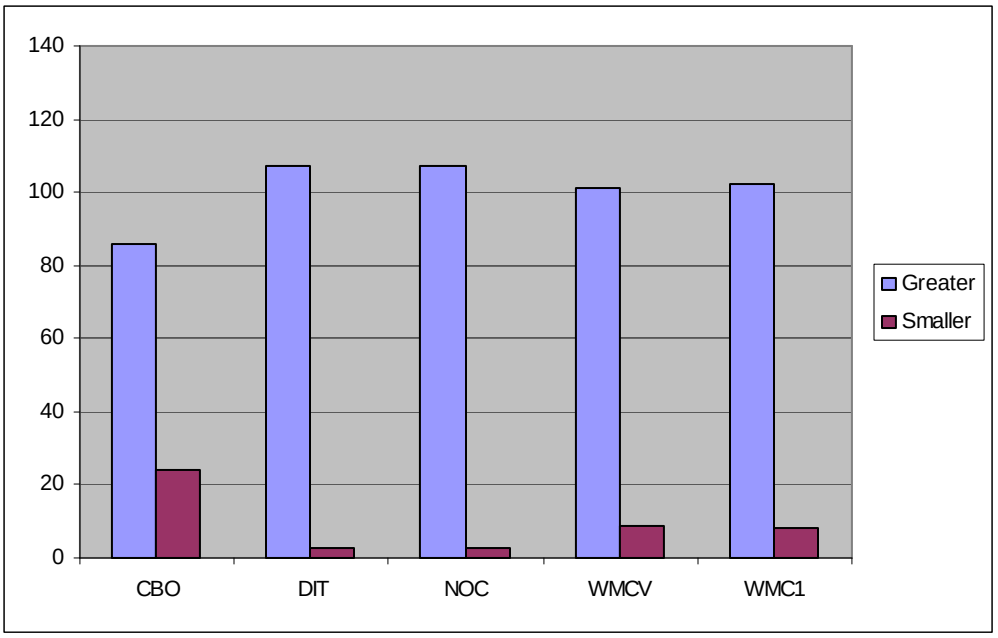

Figure 5.12 Object Oriented metric comparison for VRJuggler

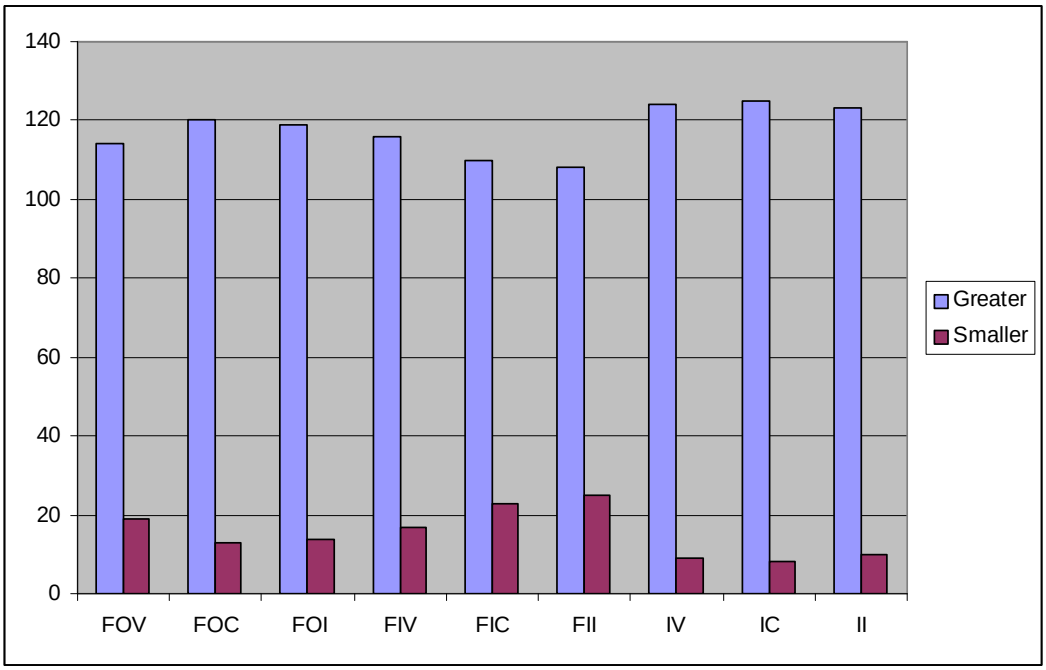

Figure 5.13 Structural metric comparison for DIVERSE

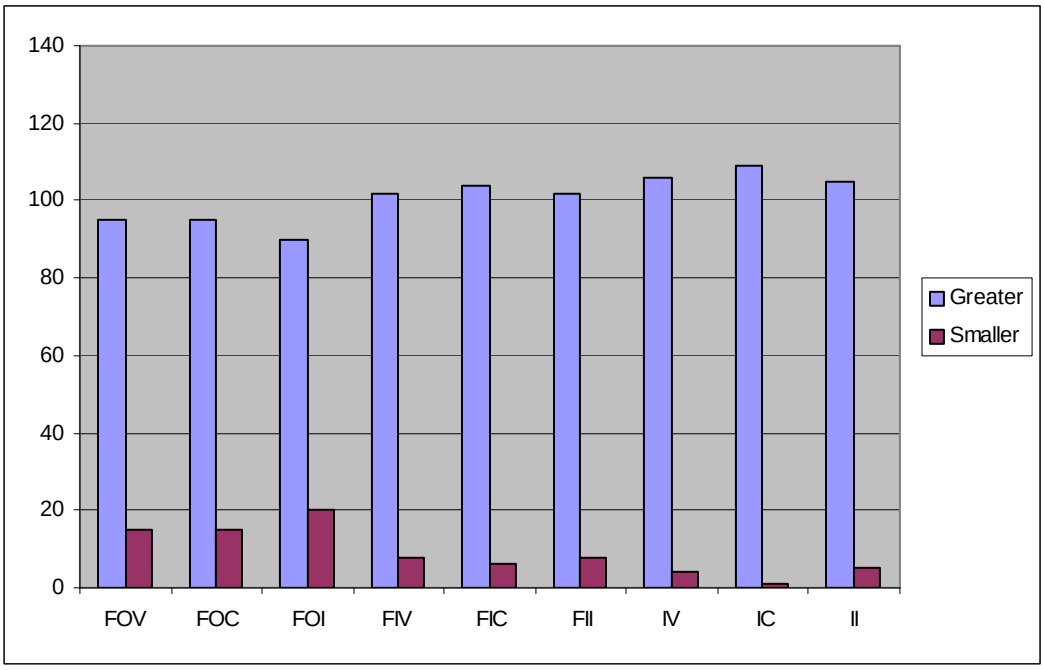

Figure 5.14 Structural metric comparison for VRJuggler

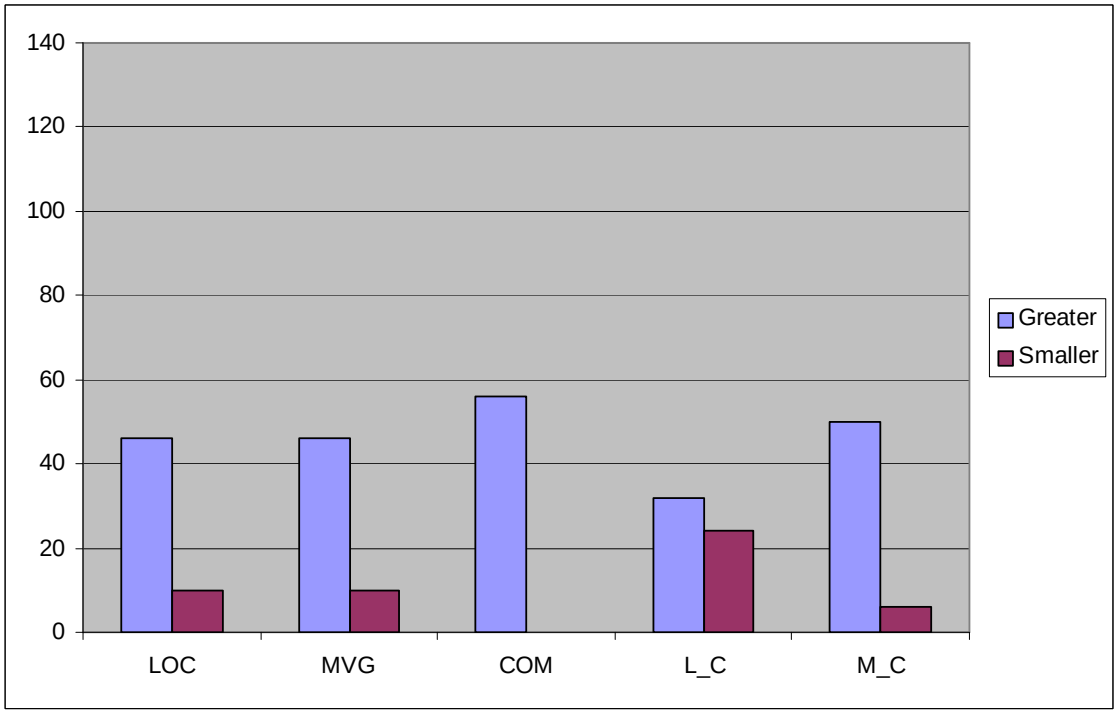

Figure 5.15 Procedural metric comparison for DIVERSE

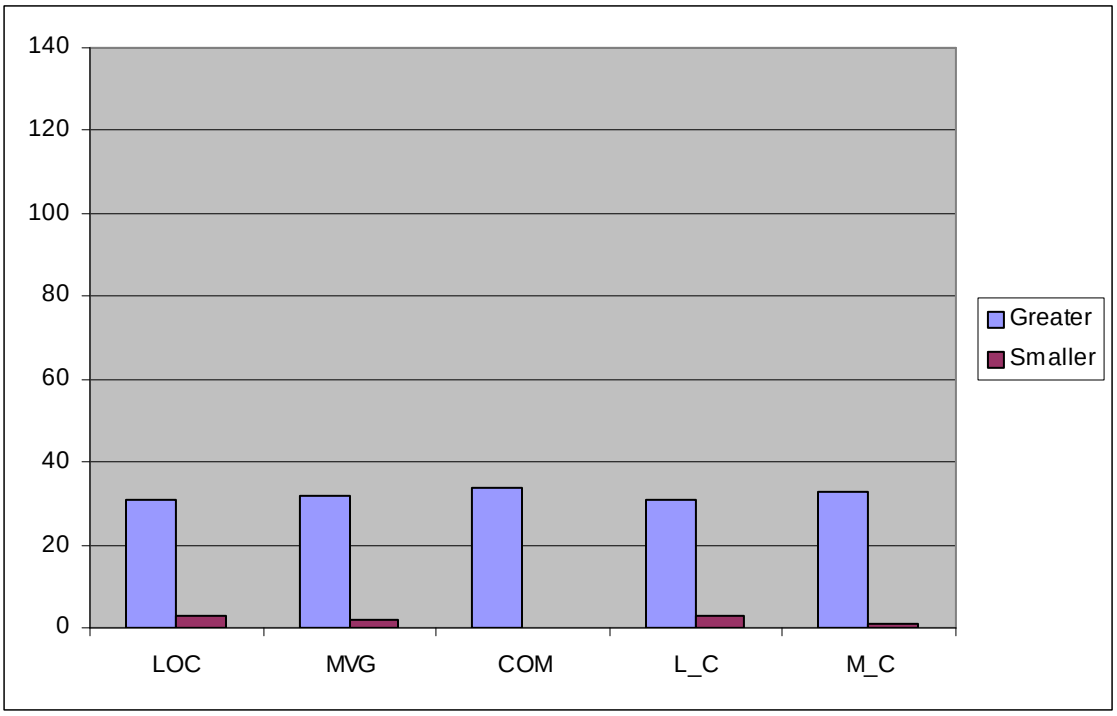

Figure 5.16 Procedural metric comparison for VRJuggler

As be can seen in all of these graphics (figure 5.11, figure 5.12, figure 5.13, figure, 5.14, figure 5.15, figure 5.16) it is clear that more classes have metrics with a higher rating in the second generation. This suggests that metric values increase as the system changes. The system changes can be adding more functionality, bug fixes, and redesign. This result was observed in other systems, and is now observed in VR systems [LIW92] [LIW93].

From a structural metrics point of view, figures 5.13 and 5.14 suggest several significant results. First complexity both within classes and in-between classes increases. This is true because as size increases more inter-connections appear. The more connected a system is, the more effort it takes to understand the implications of the connections. Testing, maintenance, and usage become more difficult.

From an object oriented point of view figures 5.11 and 5.12 suggest significant results. Inheritance hierarchies deepen between versions. Objects become more inter-

connected. More classes have children. Software evolves and takes advantage of the features of the object oriented paradigm. One interesting result is that both VRJuggler and DIVERSE show a similar response for almost every one of the metrics.

From a procedural metrics point of view figures 5.15 and 5.16 suggest that even though the change between metric groups is not as strong as the other two groups, the metric results still do grow between versions. The reason why the response is lower than with the other two paradigms is because the two systems studied were mainly object oriented in nature. This result suggests that the response from an object-oriented system is going to be stronger with object oriented metrics than with procedural metrics.

These results once again suggest the link between structural metrics and object oriented metrics. The responses from these two categories are not exactly identical but they both respond strongly. These results also show that object oriented and structural metrics are not related to procedural metrics due to the different response levels.

One last point worth considering is the effect that a general increase in program size might have on this comparative analysis. Because LOC has often been positively correlated with other procedural metrics, one might ask if the increase in code size alone might be responsible for the differences noted here. However, in this study the only metrics that are highly correlated (greater than 60%) with LOC are MVG, COM, WMC1, WMVC, and their derivative metrics. In addition to these metrics, FI* and CBO are the only other metrics that correlate with LOC in the entire study, but only in a few releases rather than study-wide. The latter two correlations are also under 30%. Thus, while an increase in LOC may influence the trends shown in Figures 5.15 and 5.16, the correlations suggest that other factors likely play a significant role in the remaining increases noted here, particularly among many of the object-oriented and structural metrics.

Chapter VI: Conclusions and Future Work

Discussion of Results

There are three results that this research has produced. They are all reaffirmations of previous results for metrics. The innovation behind the results is that it is the first time they have applied to VR. The three results are the correlation between McCabe's cyclomatic complexity and lines of code, the correlation between structural and object oriented metrics, and that metric responses grow between version levels of software.

The McCabe's cyclomatic complexity and lines of code result suggests that as the lines of code for VR software grows, the complexity of that code increases also. Even though common sense tells us this, it is beneficial to have scientific backing for common sense. From this knowledge, ideas and practices from the different metric activities can be put into place to help combat the increasing complexity of software [HENRYS81b]. For example, if a team of VR developers collects enough data from previous projects, they can use the cyclomatic complexity metric for evaluation purposes. The cyclomatic complexity metric can be used to determine when a project is reaching the complexity threshold that will start producing problems for the development team. This can help VR software development teams learn to identify and correct problems with their codebase.

The correlation between structural metrics and object oriented metrics suggests the knowledge that these two metrics work well together. This means that developers can use their results to help analyze the relationships between classes, and to help pinpoint specific problem areas of the software product in VR. Using these metrics class designs can be studied and problems will be identified easier than without metrics. For example, using previously collected data, a team of VR developers could determine when their project is reaching a threshold for too much interconnection. This could save the team from having to go through decoupling and redesigning classes to reduce complexity so that further development can proceed.

The last result from this research suggests that metric values increase between version levels. This means that over time, software gets larger, more complex, and more difficult to maintain and enhance. This result suggests the fact that software metrics really do work in VR. We know that over time software gets more complex. It gains complexity both in terms of size and dependencies, and can be shown in VR toolkits. We can measure this change and can make more educated decisions on how best to further develop software.

Conclusions

From the results in chapter 4, it can be suggested that software metrics work in VR. From the reaffirmation of the correlation between McCabe's cyclomatic complexity and lines of code to the result that metrics values increase from version to version there is a potential case that metrics work with VR toolkits. What this means for VR toolkits is that development could be improved by incorporating metrics into the software design life cycle because metrics behavior with VR toolkit is similar to other domains. Other domains have benefited from metrics, and so could VR toolkits. This addresses the initial problem of investigating whether or not knowledge from other disciplines can be applied to the development of VR software.

Contribution

The contribution of this work can be concluded that problems in VR software have been identified, and solutions have been proposed. One of these possible solutions is applying metrics to the development cycle of VR toolkits to improve their quality. This solution has been investigated and it can be suggested because metrics behave the same way that they do in other domains. Behavior in this case is defined as similar statistically significant correlations.

Future work

The discovery of suggesting that metrics do work in VR opens a whole realm of future work. There are three main areas that future work could proceed. The first area is a more in-depth study of metrics in VR. The second area would be metric discovery in

VR. The third area would be studies of metric applications to VR projects. Each of these areas could lead to important and new discoveries for the field of VR.

The first area of work could encompass a more comprehensive study involving all of the OO metrics that Chidamber created. This would provide a complete view of the results for study. In order to do this, a better metrics tool would have to be developed to rerun the study. This tool could also look at implementing other metrics that are not included in the tool used for this research. Comparisons of closed source and some of the other VR API's could also be done. Metrics could be applied to the graphical toolkits in VR. Perhaps these results could shed light on how to redesign them to make them more friendly for VR toolkit API's. One last area of study is on the processes used to develop VR API's. Would traditional software engineering methodologies apply well to VR API development? Would the need for agile models such as extreme programming need to be applied to handle the vast changes that are happening in the field of VR? Could VR API's be part of the process of merging the usability and software engineering life cycles?

The second area of study would be the investigation of metric discovery within VR. Are there specific patterns in software that only apply to VR? Are there specific metric responses that appear only in VR? Are there special types of highly specific metrics that are only present in VR? Can studying VR software discover new metrics? Would it be possible to tie software metrics into the realm of HCI and topics such as usability? Imagine the benefits of being able to quantify usability by code metrics. Perhaps this is a far-fetched dream, but it may be possible.

The third area of study would be to investigate how metrics improve the quality of a VR toolkit API. This would entail integrating metrics collection and analysis into the VR development process. This would allow for the benefit of being able to predict problems before they become problems. Quality could partially be measured by metric reports. Decisions can be influenced based on quantifiable data instead of on the gut reaction of senior software engineers. The real test in this area would be to see if given

two equal software teams assigned the task of developing a VR API and one used metrics and the other did not. This test would prove whether or not using metrics in VR does help speed the development of VR API's, along with helping increase their quality.

One last area that could be investigated is the creation and maintenance of a team of developers necessary to develop VR software. There are several different specialties that have to be harnessed to build a VR API. Software engineers, usability engineers, computer graphic specialists, and sometimes computer engineers are just some of the disciplines necessary to create a successful VR API. Learning how to effectively coordinate the diversity of people that come from these disciplines could be a very difficult task.

Bibliography

[ALBRECHTAJ83] Albrecht A.J., & Gaafney, J.E. Software function, source lines of code and development effort prediction: a software science validation. IEEE Transactions on Software Engineering, SE-9(6), 639-6648, November 1983.

[BIERBAUMA00] A. Bierbaum, C. Just, P. Hartling, and C. Cruz-Neira. Flexible application design using vrjuggler. In Conference Abstracts and Applications. SIGGRAPH, 2000.

[BIERBAUMA01] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vrjuggler: A virtual platform for virual reality application development. In IEEE VR 2001.

[BOWMAN01] Virtual Environments Course Website http://courses.cs.vt.edu/~cs5754 (April 04)

[BROOKSFPS99] F.P. Brooks What's Real About Virtual Reality? IEEE Computer Graphics and Applications 1999

[CARLSSONC93] Carlsson, C. and Hagsand, O., "DIVE - A Multi-User Virtual Reality System," Proc. of IEEE VRAIS "93, Seattle, WA, September t8-22, 1993, pp. 394-400.

[CHIDAMBERSR91] Chidamber, S. R., & Kemerer, C.F. (1991). Towards a metric suite for object-oriented design. In OOPSLA '91: Conference on object-oriented programming, systems, languages and applications(pp. 184-196). New York: ACM Press.

[CRUZC93] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen projection based virtual reality: The design and implementation of the cave. In Computer Graphics. SIGGRAPH, 1993.

[DOD04] DOD-STD-2167A http://tecnet.jcte.jcs.mil/htdocs/teinfo/software/did25.html (March 2004)

[GAMMAE05] Erich Gamma, Richard Helm, Ralph Jonnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, Massachusetts: Addison-Wesley Publishing Company

[HALSTEDM77] Halstead, Maurice H., "Elements of Software Science," *Elsvier North-Holland*, New York, 1977

[HENRYS81a]Henry, S.M. and Kafura, D.G. Software Structure Metrics Based on Information Flow,IEEE Transactions on Software Engineering, (7,5), September, 1981, pp. 510 – 518

[HENRYS81b] Henry, S.M., Kafura, D.G. and Harris, K., On the Relationships Among Three Software Metrics, Proceedings of the 1981 ACM SIGMETRICS Symposium on Measurement and Evaluation of Software Quality,March, 1981, pp. 81-88.
November, 1993.
[HENRYS84] Henry, S., & Kafura, D. (1984) The evaluation of software systems structure using quantitative software metrics. Software Practice and Experience, 14(6), 561-573

[HENRYS91] Henry, S.M., Wake, S.A., "Predicting Maintainability with Software Quality Metrics", Journal of Software Maintenance, September, 1991, pp. 129-143.

[JOHNSONA98] Johnson, A. E., Leigh, J., and DeFanti T., "Multi- Disciplinary Experiences with CAVERNsoft Tele-Immersive Applications," Proc. of Fourth International Conference on Virtual System and Multimedia, November 1998, pp. 498-503.

[KELSOJ02] Kelso, J., Arsenault, L., Kriz, R., and Satterfield, S. DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments. To appear in the Proceedings of IEEE Virtual Reality, 2002.

[KESSLERG00] Kessler, G., Bowman, D., and Hodges, L. The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications. Presence: Teleoperators and Virtual Environments, vol. 9, no. 2, 2000, pp. 187-208.

[LEIGHJ97] Leigh, J., Johnson, A. E. and DeFanti, T. A., "CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments," Journal of Virtual Reality Research, Development and Applications, 2(2):217-237, 1997:

[LITTLEFAIRT01] Tim Littlefair. An Investigation into the use of software code metrics in the industrial software development. Ph.D. dissertation, Edith Cowan University, 2001

[LIW92] Li Wei, "Applying Software Maintenance Metrics in the Object Oriented Software Development Lifecycle," *Ph.D. Dissertation*, Virginia Polytechnic Institute and State University, Department of Computer Science, September 1991

[LIW93] Li, W. Henry, S. (1993) Object-oriented metrics that predict maintainability. Journal of Systems and Software, Vol. 23, No. 2.

[MCCABETJ76] McCabe, T.J. A Complexity measure. IEEE Transactions on Software Engineering, SE-2(4), 308-320, December 1976.

[MCCABETJ89] McCabe, Thomas J. and C. Butler, "Design Complexity Measurement and Testing," *Communications of the ACM*, December 1989, pp. 1415-1424.

[OLSONE02] E. Olson. Cluster Juggler - PC Cluster Virtual Reality. M.Sc. thesis, Iowa State University, 2002.

[PARKK00] K. Park et al., CAVERNsoft G2: A Toolkit for High Performance Tele-immersive Collaboration, Proc. ACM Symp. Virtual Reality Software and Technology, ACM Press, New York, 2000, pp. 8-15

[PAULKM93] Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, capability Maturity Model, Version 1.1, IEEE Software, Vol. 10, No. 4, July 1993

[POLYSN04] Polys, N., North, C., Bowman, D., Ray, A., Moldenhauer, M., and Dandekar, C. Snap2Diverse: Coordinating Information Visualizations and Virtual Environments. Proceedings of SPIE Visualization and Data Analysis, 2004.

[SAS01] SAS Website http://www.sas.com (March 2004)

[SCHAFFERB03] Benjamin Schaeffer and Camille Goudeseune. Syzygy: Native PC Cluster VR. IEEE Virtual Reality Conference 2003.

[SCHAFFERB04] Benjamin Schaffer Networking and Management Frameworks for Cluster-Based Graphics http://www.isl.uiuc.edu/Publications/ClusterGraphics.pdf (February 2004)

Appendix: Correlations Between Metrics

This appendix contains the raw correlation results from the statistical consulting center. These are the results from running the metrics tool against the source code, formatting the output of the metrics tool, and then running SAS against the formatted output. These results are in a tabular format where the metric on the left column is compared to the metric on the top row. Each of these pairings has three numbers associated with it. The first number is the percentage of significance. The second number is the p value for the correlation. The third number is the number of observations taken to determine the previous two numbers.

The data is organized by each version of the software starting with DIVERSE and ending with VRJuggler.

Correlations between metrics for Diverse1

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
WMC1	1.00000	0.83983	-0.02149	0.03193	0.24420	0.92769	0.92686
		<.0001	0.7609	0.6511	0.0004	<.0001	<.0001
	203	203	203	203	203	203	203
WMCV	0.83983	1.00000	0.06953	0.08171	0.48780	0.61145	0.61474
	<.0001		0.3243	0.2465	<.0001	<.0001	<.0001
	203	203	203	203	203	203	203
DIT	-0.02149	0.06953	1.00000	-0.05807	0.06159	-0.02089	-0.02958
	0.7609	0.3243		0.4105	0.3827	0.7673	0.6753
	203	203	203	203	203	203	203

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
NOC	0.03193	0.08171	-0.05807	1.00000	0.49903	-0.01361	-0.01548
	0.6511	0.2465	0.4105		<.0001	0.8472	0.8265
	203	203	203	203	203	203	203
CBO	0.24420	0.48780	0.06159	0.49903	1.00000	0.08120	0.08658
	0.0004	<.0001	0.3827	<.0001		0.2495	0.2194
	203	203	203	203	203	203	203
LOC	0.92769	0.61145	-0.02089	-0.01361	0.08120	1.00000	0.99748
	<.0001	<.0001	0.7673	0.8472	0.2495		<.0001
	203	203	203	203	203	203	203
MVG	0.92686	0.61474	-0.02958	-0.01548	0.08658	0.99748	1.00000
	<.0001	<.0001	0.6753	0.8265	0.2194	<.0001	
	203	203	203	203	203	203	203
COM	0.91429	0.64742	0.00627	0.05786	0.17375	0.97124	0.96289
	<.0001	<.0001	0.9292	0.4122	0.0132	<.0001	<.0001
	203	203	203	203	203	203	203
L_C	-0.03531	-0.06344	0.22377	-0.07618	-0.05087	0.00045	0.00342
	0.6788	0.4564	0.0079	0.3710	0.5506	0.9958	0.9680
	140	140	140	140	140	140	140
M_C	-0.00374	-0.02026	0.21289	-0.07515	-0.02823	0.02767	0.03844
	0.9667	0.8212	0.0163	0.4011	0.7527	0.7575	0.6679
	127	127	127	127	127	127	127
FOV	0.10489	0.21639	-0.13788	0.66827	0.79899	0.02703	0.03427
	0.1364	0.0019	0.0498	<.0001	<.0001	0.7019	0.6274
	203	203	203	203	203	203	203
FOC	-0.00479	0.02302	-0.16434	0.89530	0.51994	-0.03260	-0.03379
	0.9460	0.7445	0.0191	<.0001	<.0001	0.6443	0.6322
	203	203	203	203	203	203	203

FOI	0.19084	0.37833	-0.14802	0.56488	0.93598	0.05308	0.06099
	0.0064	<.0001	0.0351	<.0001	<.0001	0.4519	0.3874
	203	203	203	203	203	203	203
FIV	0.29832	0.59795	0.47860	0.01292	0.48271	0.08516	0.08887
	<.0001	<.0001	<.0001	0.8549	<.0001	0.2270	0.2074
	203	203	203	203	203	203	203
	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
FIC	0.18185	0.39408	0.60339	0.05631	0.34191	0.09156	0.07984
	0.0094	<.0001	<.0001	0.4249	<.0001	0.1939	0.2575
	203	203	203	203	203	203	203
FII	0.19623	0.39942	0.55978	-0.05403	0.40153	0.09222	0.08692
	0.0050	<.0001	<.0001	0.4439	<.0001	0.1906	0.2175
	203	203	203	203	203	203	203
IV	0.10976	0.20557	-0.02253	0.04098	0.57883	0.05430	0.06452
	0.1190	0.0033	0.7496	0.5615	<.0001	0.4416	0.3604
	203	203	203	203	203	203	203
IC	0.02205	0.05174	-0.02763	0.87757	0.44744	-0.00982	-0.01180
	0.7549	0.4635	0.6955	<.0001	<.0001	0.8894	0.8673
	203	203	203	203	203	203	203
II	0.26063	0.46580	-0.02597	0.01230	0.66695	0.10676	0.11187
	0.0002	<.0001	0.7130	0.8617	<.0001	0.1295	0.1121
	203	203	203	203	203	203	203

	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMC1	0.91429	-0.03531	-0.00374	0.10489	-0.00479	0.19084	0.29832
	<.0001	0.6788	0.9667	0.1364	0.9460	0.0064	<.0001
	203	140	127	203	203	203	203
WMCV	0.64742	-0.06344	-0.02026	0.21639	0.02302	0.37833	0.59795
	<.0001	0.4564	0.8212	0.0019	0.7445	<.0001	<.0001
	203	140	127	203	203	203	203
DIT	0.00627	0.22377	0.21289	-0.13788	-0.16434	-0.14802	0.47860
	0.9292	0.0079	0.0163	0.0498	0.0191	0.0351	<.0001
	203	140	127	203	203	203	203
NOC	0.05786	-0.07618	-0.07515	0.66827	0.89530	0.56488	0.01292
	0.4122	0.3710	0.4011	<.0001	<.0001	<.0001	0.8549
	203	140	127	203	203	203	203
	COM	L_C	M_C	FOV	FOC	FOI	FIV
CBO	0.17375	-0.05087	-0.02823	0.79899	0.51994	0.93598	0.48271
	0.0132	0.5506	0.7527	<.0001	<.0001	<.0001	<.0001
	203	140	127	203	203	203	203
LOC	0.97124	0.00045	0.02767	0.02703	-0.03260	0.05308	0.08516
	<.0001	0.9958	0.7575	0.7019	0.6443	0.4519	0.2270
	203	140	127	203	203	203	203
MVG	0.96289	0.00342	0.03844	0.03427	-0.03379	0.06099	0.08887
	<.0001	0.9680	0.6679	0.6274	0.6322	0.3874	0.2074
	203	140	127	203	203	203	203
COM	1.00000	-0.07471	-0.05235	0.11471	0.02685	0.13610	0.14216
		0.3804	0.5589	0.1032	0.7038	0.0529	0.0430
	203	140	127	203	203	203	203
L_C	-0.07471	1.00000	0.98533	-0.07901	-0.08063	-0.09856	0.02833
	0.3804		<.0001	0.3534	0.3436	0.2467	0.7397
	140	140	127	140	140	140	140
M_C	-0.05235	0.98533	1.00000	-0.06973	-0.08047	-0.08648	0.00506
	0.5589	<.0001		0.4360	0.3685	0.3337	0.9550
	127	127	127	127	127	127	127
FOV	0.11471	-0.07901	-0.06973	1.00000	0.67834	0.87984	0.13446
	0.1032	0.3534	0.4360		<.0001	<.0001	0.0558
	203	140	127	203	203	203	203
FOC	0.02685	-0.08063	-0.08047	0.67834	1.00000	0.62247	-0.08610
	0.7038	0.3436	0.3685	<.0001		<.0001	0.2219
	203	140	127	203	203	203	203
FOI	0.13610	-0.09856	-0.08648	0.87984	0.62247	1.00000	0.25197
	0.0529	0.2467	0.3337	<.0001	<.0001		0.0003
	203	140	127	203	203	203	203
FIV	0.14216	0.02833	0.00506	0.13446	-0.08610	0.25197	1.00000
	0.0430	0.7397	0.9550	0.0558	0.2219	0.0003	
	203	140	127	203	203	203	203

FIC	0.15848	0.12609	0.13652	0.03005	-0.04057	0.07029	0.71241
	0.0239	0.1377	0.1259	0.6704	0.5655	0.3190	<.0001
	203	140	127	203	203	203	203
	COM	L_C	M_C	FOV	FOC	FOI	FIV
FII	0.13878	0.14223	0.19566	-0.02260	-0.14455	0.05341	0.71371
	0.0483	0.0937	0.0275	0.7490	0.0396	0.4492	<.0001
	203	140	127	203	203	203	203
IV	0.11832	-0.02852	-0.02133	0.69665	0.02954	0.59403	0.22795
	0.0927	0.7380	0.8119	<.0001	0.6756	<.0001	0.0011
	203	140	127	203	203	203	203
IC	0.04278	-0.04746	-0.05053	0.58809	0.79232	0.49884	0.00939
	0.5445	0.5776	0.5726	<.0001	<.0001	<.0001	0.8942
	203	140	127	203	203	203	203
II	0.14923	-0.02609	-0.01608	0.36702	0.00052	0.61852	0.36148
	0.0336	0.7596	0.8576	<.0001	0.9941	<.0001	<.0001
	203	140	127	203	203	203	203

	FIC	FII	IV	IC	II
WMC1	0.18185	0.19623	0.10976	0.02205	0.26063
	0.0094	0.0050	0.1190	0.7549	0.0002
	203	203	203	203	203
WMCV	0.39408	0.39942	0.20557	0.05174	0.46580
	<.0001	<.0001	0.0033	0.4635	<.0001
	203	203	203	203	203
DIT	0.60339	0.55978	-0.02253	-0.02763	-0.02597
	<.0001	<.0001	0.7496	0.6955	0.7130
	203	203	203	203	203
NOC	0.05631	-0.05403	0.04098	0.87757	0.01230
	0.4249	0.4439	0.5615	<.0001	0.8617
	203	203	203	203	203
CBO	0.34191	0.40153	0.57883	0.44744	0.66695
	<.0001	<.0001	<.0001	<.0001	<.0001
	203	203	203	203	203
	FIC	FII	IV	IC	II
LOC	0.09156	0.09222	0.05430	-0.00982	0.10676
	0.1939	0.1906	0.4416	0.8894	0.1295
	203	203	203	203	203
MVG	0.07984	0.08692	0.06452	-0.01180	0.11187
	0.2575	0.2175	0.3604	0.8673	0.1121
	203	203	203	203	203
COM	0.15848	0.13878	0.11832	0.04278	0.14923
	0.0239	0.0483	0.0927	0.5445	0.0336
	203	203	203	203	203
L_C	0.12609	0.14223	-0.02852	-0.04746	-0.02609
	0.1377	0.0937	0.7380	0.5776	0.7596
	140	140	140	140	140
M_C	0.13652	0.19566	-0.02133	-0.05053	-0.01608
	0.1259	0.0275	0.8119	0.5726	0.8576
	127	127	127	127	127
FOV	0.03005	-0.02260	0.69665	0.58809	0.36702
	0.6704	0.7490	<.0001	<.0001	<.0001
	203	203	203	203	203
FOC	-0.04057	-0.14455	0.02954	0.79232	0.00052
	0.5655	0.0396	0.6756	<.0001	0.9941
	203	203	203	203	203
FOI	0.07029	0.05341	0.59403	0.49884	0.61852
	0.3190	0.4492	<.0001	<.0001	<.0001
	203	203	203	203	203
FIV	0.71241	0.71371	0.22795	0.00939	0.36148
	<.0001	<.0001	0.0011	0.8942	<.0001
	203	203	203	203	203
FIC	1.00000	0.78698	0.05660	0.05467	0.13728
		<.0001	0.4225	0.4385	0.0508
	203	203	203	203	203
FII	0.78698	1.00000	0.09648	-0.02856	0.28272
	<.0001		0.1709	0.6859	<.0001
	203	203	203	203	203
	FIC	FII	IV	IC	II

IV	0.05660	0.09648	1.00000	0.05233	0.51807
	0.4225	0.1709		0.4584	<.0001
	203	203	203	203	203
IC	0.05467	-0.02856	0.05233	1.00000	0.02195
	0.4385	0.6859	0.4584		0.7559
	203	203	203	203	203
II	0.13728	0.28272	0.51807	0.02195	1.00000
	0.0508	<.0001	<.0001	0.7559	
	203	203	203	203	203

Correlations between metrics for Diverse2

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
WMC1	1.00000	0.76401	0.01466	0.00257	0.19447	0.94998	0.94611
		<.0001	0.8273	0.9695	0.0035	<.0001	<.0001
	224	224	224	224	224	224	224
WMCV	0.76401	1.00000	0.25696	-0.05829	0.28413	0.59022	0.57788
	<.0001		0.0001	0.3852	<.0001	<.0001	<.0001
	224	224	224	224	224	224	224
DIT	0.01466	0.25696	1.00000	-0.11352	0.11097	-0.01045	-0.01956
	0.8273	0.0001		0.0901	0.0976	0.8764	0.7710
	224	224	224	224	224	224	224
	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
NOC	0.00257	-0.05829	-0.11352	1.00000	0.45235	-0.01973	-0.01960
	0.9695	0.3852	0.0901		<.0001	0.7690	0.7705
	224	224	224	224	224	224	224
CBO	0.19447	0.28413	0.11097	0.45235	1.00000	0.06643	0.06774
	0.0035	<.0001	0.0976	<.0001		0.3223	0.3128
	224	224	224	224	224	224	224
LOC	0.94998	0.59022	-0.01045	-0.01973	0.06643	1.00000	0.99834
	<.0001	<.0001	0.8764	0.7690	0.3223		<.0001
	224	224	224	224	224	224	224
MVG	0.94611	0.57788	-0.01956	-0.01960	0.06774	0.99834	1.00000
	<.0001	<.0001	0.7710	0.7705	0.3128	<.0001	
	224	224	224	224	224	224	224
COM	0.93555	0.58631	-0.02040	-0.01372	0.03691	0.98916	0.98243
	<.0001	<.0001	0.7613	0.8381	0.5827	<.0001	<.0001
	224	224	224	224	224	224	224
L_C	-0.02534	-0.05479	0.25874	-0.05831	0.00151	0.00274	0.01160
	0.7688	0.5248	0.0023	0.4985	0.9861	0.9747	0.8929
	137	137	137	137	137	137	137
M_C	-0.02048	-0.06590	0.13671	-0.04974	-0.04414	-0.00083	0.01576
	0.8251	0.4764	0.1382	0.5912	0.6336	0.9929	0.8649
	119	119	119	119	119	119	119
FOV	0.03865	-0.06558	-0.17644	0.69723	0.68185	0.00377	0.00789
	0.5650	0.3285	0.0081	<.0001	<.0001	0.9552	0.9065
	224	224	224	224	224	224	224
FOC	0.00786	-0.09940	-0.21569	0.84844	0.50740	-0.00624	-0.00144
	0.9068	0.1381	0.0012	<.0001	<.0001	0.9260	0.9829
	224	224	224	224	224	224	224
FOI	0.12502	0.07616	-0.24111	0.54693	0.86866	0.04060	0.04634
	0.0618	0.2563	0.0003	<.0001	<.0001	0.5455	0.4902
	224	224	224	224	224	224	224
FIV	0.26744	0.56284	0.56685	-0.04217	0.45237	0.07838	0.07310
	<.0001	<.0001	<.0001	0.5301	<.0001	0.2427	0.2760
	224	224	224	224	224	224	224
	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
FIC	0.11585	0.33478	0.71647	-0.05691	0.33021	0.04050	0.03157
	0.0836	<.0001	<.0001	0.3966	<.0001	0.5465	0.6384
	224	224	224	224	224	224	224
FII	0.16661	0.43544	0.65840	-0.07422	0.44968	0.06071	0.05301
	0.0125	<.0001	<.0001	0.2686	<.0001	0.3658	0.4298
	224	224	224	224	224	224	224
IV	0.12194	-0.00224	-0.06967	0.16268	0.41657	0.06183	0.06861
	0.0685	0.9734	0.2992	0.0148	<.0001	0.3570	0.3066
	224	224	224	224	224	224	224
IC	0.06111	-0.02637	-0.07084	0.89015	0.48755	0.03128	0.03734
	0.3626	0.6947	0.2911	<.0001	<.0001	0.6415	0.5782
	224	224	224	224	224	224	224

II	0.25161	0.52126	0.05721	0.00240	0.45561	0.09918	0.08937
	0.0001	<.0001	0.3941	0.9716	<.0001	0.1389	0.1826
	224	224	224	224	224	224	224
	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMC1	0.93555	-0.02534	-0.02048	0.03865	0.00786	0.12502	0.26744
	<.0001	0.7688	0.8251	0.5650	0.9068	0.0618	<.0001
	224	137	119	224	224	224	224
WMCV	0.58631	-0.05479	-0.06590	-0.06558	-0.09940	0.07616	0.56284
	<.0001	0.5248	0.4764	0.3285	0.1381	0.2563	<.0001
	224	137	119	224	224	224	224
DIT	-0.02040	0.25874	0.13671	-0.17644	-0.21569	-0.24111	0.56685
	0.7613	0.0023	0.1382	0.0081	0.0012	0.0003	<.0001
	224	137	119	224	224	224	224
NOC	-0.01372	-0.05831	-0.04974	0.69723	0.84844	0.54693	-0.04217
	0.8381	0.4985	0.5912	<.0001	<.0001	<.0001	0.5301
	224	137	119	224	224	224	224
	COM	L_C	M_C	FOV	FOC	FOI	FIV
CBO	0.03691	0.00151	-0.04414	0.68185	0.50740	0.86866	0.45237
	0.5827	0.9861	0.6336	<.0001	<.0001	<.0001	<.0001
	224	137	119	224	224	224	224
LOC	0.98916	0.00274	-0.00083	0.00377	-0.00624	0.04060	0.07838
	<.0001	0.9747	0.9929	0.9552	0.9260	0.5455	0.2427
	224	137	119	224	224	224	224
MVG	0.98243	0.01160	0.01576	0.00789	-0.00144	0.04634	0.07310
	<.0001	0.8929	0.8649	0.9065	0.9829	0.4902	0.2760
	224	137	119	224	224	224	224
COM	1.00000	-0.05350	-0.05624	0.00340	-0.01321	0.02559	0.03631
		0.5346	0.5435	0.9597	0.8442	0.7032	0.5888
	224	137	119	224	224	224	224
L_C	-0.05350	1.00000	0.95138	-0.06655	-0.05101	-0.06321	0.10006
	0.5346		<.0001	0.4397	0.5539	0.4631	0.2447
	137	137	119	137	137	137	137
M_C	-0.05624	0.95138	1.00000	-0.05965	-0.05213	-0.06400	0.03750
	0.5435	<.0001		0.5193	0.5734	0.4893	0.6856
	119	119	119	119	119	119	119
FOV	0.00340	-0.06655	-0.05965	1.00000	0.69461	0.82753	-0.04534
	0.9597	0.4397	0.5193		<.0001	<.0001	0.4996
	224	137	119	224	224	224	224
FOC	-0.01321	-0.05101	-0.05213	0.69461	1.00000	0.65252	-0.10750
	0.8442	0.5539	0.5734	<.0001		<.0001	0.1086
	224	137	119	224	224	224	224
FOI	0.02559	-0.06321	-0.06400	0.82753	0.65252	1.00000	0.08611
	0.7032	0.4631	0.4893	<.0001	<.0001		0.1992
	224	137	119	224	224	224	224
FIV	0.03631	0.10006	0.03750	-0.04534	-0.10750	0.08611	1.00000
	0.5888	0.2447	0.6856	0.4996	0.1086	0.1992	
	224	137	119	224	224	224	224
FIC	0.01191	0.19835	0.08858	-0.09655	-0.13788	-0.07515	0.71393
	0.8593	0.0202	0.3380	0.1498	0.0392	0.2627	<.0001
	224	137	119	224	224	224	224
	COM	L_C	M_C	FOV	FOC	FOI	FIV
FII	0.02825	0.13447	0.03539	-0.11749	-0.15362	-0.05187	0.75664
	0.6741	0.1172	0.7024	0.0793	0.0215	0.4398	<.0001
	224	137	119	224	224	224	224
IV	0.02103	0.01462	0.01612	0.53737	0.14158	0.44803	0.12809
	0.7543	0.8653	0.8619	<.0001	0.0342	<.0001	0.0556
	224	137	119	224	224	224	224
IC	0.01537	-0.03815	-0.04159	0.63856	0.81226	0.55548	0.02378
	0.8190	0.6581	0.6534	<.0001	<.0001	<.0001	0.7234
	224	137	119	224	224	224	224
II	0.07320	-0.02333	-0.03187	0.05926	0.00862	0.32649	0.41858
	0.2753	0.7867	0.7308	0.3774	0.8979	<.0001	<.0001
	224	137	119	224	224	224	224

	FIC	FII	IV	IC	II
WMC1	0.11585	0.16661	0.12194	0.06111	0.25161
	0.0836	0.0125	0.0685	0.3626	0.0001
	224	224	224	224	224
WMCV	0.33478	0.43544	-0.00224	-0.02637	0.52126

	<.0001	<.0001	0.9734	0.6947	<.0001
	224	224	224	224	224
DIT	0.71647	0.65840	-0.06967	-0.07084	0.05721
	<.0001	<.0001	0.2992	0.2911	0.3941
	224	224	224	224	224
NOC	-0.05691	-0.07422	0.16268	0.89015	0.00240
	0.3966	0.2686	0.0148	<.0001	0.9716
	224	224	224	224	224
CBO	0.33021	0.44968	0.41657	0.48755	0.45561
	<.0001	<.0001	<.0001	<.0001	<.0001
	224	224	224	224	224

	FIC	FII	IV	IC	II
LOC	0.04050	0.06071	0.06183	0.03128	0.09918
	0.5465	0.3658	0.3570	0.6415	0.1389
	224	224	224	224	224
MVG	0.03157	0.05301	0.06861	0.03734	0.08937
	0.6384	0.4298	0.3066	0.5782	0.1826
	224	224	224	224	224
COM	0.01191	0.02825	0.02103	0.01537	0.07320
	0.8593	0.6741	0.7543	0.8190	0.2753
	224	224	224	224	224
L_C	0.19835	0.13447	0.01462	-0.03815	-0.02333
	0.0202	0.1172	0.8653	0.6581	0.7867
	137	137	137	137	137
M_C	0.08858	0.03539	0.01612	-0.04159	-0.03187
	0.3380	0.7024	0.8619	0.6534	0.7308
	119	119	119	119	119
FOV	-0.09655	-0.11749	0.53737	0.63856	0.05926
	0.1498	0.0793	<.0001	<.0001	0.3774
	224	224	224	224	224
FOC	-0.13788	-0.15362	0.14158	0.81226	0.00862
	0.0392	0.0215	0.0342	<.0001	0.8979
	224	224	224	224	224
FOI	-0.07515	-0.05187	0.44803	0.55548	0.32649
	0.2627	0.4398	<.0001	<.0001	<.0001
	224	224	224	224	224
FIV	0.71393	0.75664	0.12809	0.02378	0.41858
	<.0001	<.0001	0.0556	0.7234	<.0001
	224	224	224	224	224
FIC	1.00000	0.80115	0.05481	0.01339	0.19402
		<.0001	0.4143	0.8420	0.0036
	224	224	224	224	224
FII	0.80115	1.00000	0.03196	-0.01867	0.32980
	<.0001		0.6343	0.7810	<.0001
	224	224	224	224	224

	FIC	FII	IV	IC	II
IV	0.05481	0.03196	1.00000	0.16610	0.10469
	0.4143	0.6343		0.0128	0.1182
	224	224	224	224	224
IC	0.01339	-0.01867	0.16610	1.00000	0.03100
	0.8420	0.7810	0.0128		0.6445
	224	224	224	224	224
II	0.19402	0.32980	0.10469	0.03100	1.00000
	0.0036	<.0001	0.1182	0.6445	
	224	224	224	224	224

Correlations between metrics for Diverse3

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
WMC1	1.00000	0.52555	0.06541	-0.01551	0.05955	0.65870	0.26729
		<.0001	0.3010	0.8064	0.3465	<.0001	<.0001
	252	252	252	252	252	252	252
WMCV	0.52555	1.00000	0.24883	-0.00569	0.04623	0.10931	0.06677
	<.0001		<.0001	0.9283	0.4650	0.0833	0.2911
	252	252	252	252	252	252	252
DIT	0.06541	0.24883	1.00000	-0.05354	0.02929	0.08046	-0.02385
	0.3010	<.0001		0.3973	0.6435	0.2030	0.7063
	252	252	252	252	252	252	252
	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG

NOC	-0.01551	-0.00569	-0.05354	1.00000	0.22353	-0.07097	-0.04119
	0.8064	0.9283	0.3973		0.0003	0.2617	0.5151
	252	252	252	252	252	252	252
CBO	0.05955	0.04623	0.02929	0.22353	1.00000	0.05045	0.01151
	0.3465	0.4650	0.6435	0.0003		0.4252	0.8557
	252	252	252	252	252	252	252
LOC	0.65870	0.10931	0.08046	-0.07097	0.05045	1.00000	0.76295
	<.0001	0.0833	0.2030	0.2617	0.4252		<.0001
	252	252	252	252	252	252	252
MVG	0.26729	0.06677	-0.02385	-0.04119	0.01151	0.76295	1.00000
	<.0001	0.2911	0.7063	0.5151	0.8557	<.0001	
	252	252	252	252	252	252	252
COM	0.72696	0.15448	-0.12511	-0.03807	0.08407	0.57993	0.26107
	<.0001	0.0141	0.0473	0.5474	0.1834	<.0001	<.0001
	252	252	252	252	252	252	252
L_C	-0.02591	-0.07478	0.17165	-0.02042	0.04041	0.50610	0.53609
	0.7778	0.4149	0.0598	0.8241	0.6599	<.0001	<.0001
	121	121	121	121	121	121	121
M_C	-0.07061	-0.04835	0.06781	-0.03213	-0.03827	0.61162	0.91102
	0.4829	0.6311	0.5005	0.7498	0.7040	<.0001	<.0001
	101	101	101	101	101	101	101
FOV	-0.04372	0.00933	-0.05283	0.23971	0.96275	-0.07353	-0.03999
	0.4896	0.8829	0.4037	0.0001	<.0001	0.2448	0.5274
	252	252	252	252	252	252	252
FOC	-0.04991	-0.00126	-0.05474	0.25894	0.50881	-0.07371	-0.03665
	0.4302	0.9841	0.3868	<.0001	<.0001	0.2437	0.5625
	252	252	252	252	252	252	252
FOI	-0.04081	0.01522	-0.05284	0.23689	0.96258	-0.07315	-0.04028
	0.5190	0.8100	0.4036	0.0001	<.0001	0.2473	0.5244
	252	252	252	252	252	252	252
FIV	0.37084	0.06346	0.23452	-0.04345	0.13723	0.45218	0.18794
	<.0001	0.3157	0.0002	0.4923	0.0294	<.0001	0.0027
	252	252	252	252	252	252	252

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
FIC	0.16869	0.31004	0.64007	0.01137	0.09386	0.15665	0.03582
	0.0073	<.0001	<.0001	0.8575	0.1373	0.0128	0.5714
	252	252	252	252	252	252	252
FII	0.36692	0.11355	0.30014	-0.04719	0.14374	0.45173	0.18922
	<.0001	0.0720	<.0001	0.4558	0.0225	<.0001	0.0026
	252	252	252	252	252	252	252
IV	0.17182	0.02024	0.00626	0.11929	0.17469	0.13167	0.04752
	0.0062	0.7491	0.9212	0.0586	0.0054	0.0367	0.4527
	252	252	252	252	252	252	252
IC	0.03583	0.03252	-0.02766	0.61214	0.05131	-0.02513	-0.01732
	0.5713	0.6074	0.6621	<.0001	0.4174	0.6913	0.7844
	252	252	252	252	252	252	252
II	0.19023	0.03891	0.01525	0.10883	0.18175	0.14509	0.04900
	0.0024	0.5387	0.8096	0.0847	0.0038	0.0212	0.4387
	252	252	252	252	252	252	252

	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMC1	0.72696	-0.02591	-0.07061	-0.04372	-0.04991	-0.04081	0.37084
	<.0001	0.7778	0.4829	0.4896	0.4302	0.5190	<.0001
	252	121	101	252	252	252	252
WMCV	0.15448	-0.07478	-0.04835	0.00933	-0.00126	0.01522	0.06346
	0.0141	0.4149	0.6311	0.8829	0.9841	0.8100	0.3157
	252	121	101	252	252	252	252
DIT	-0.12511	0.17165	0.06781	-0.05283	-0.05474	-0.05284	0.23452
	0.0473	0.0598	0.5005	0.4037	0.3868	0.4036	0.0002
	252	121	101	252	252	252	252
NOC	-0.03807	-0.02042	-0.03213	0.23971	0.25894	0.23689	-0.04345
	0.5474	0.8241	0.7498	0.0001	<.0001	0.0001	0.4923
	252	121	101	252	252	252	252

	COM	L_C	M_C	FOV	FOC	FOI	FIV
CBO	0.08407	0.04041	-0.03827	0.96275	0.50881	0.96258	0.13723
	0.1834	0.6599	0.7040	<.0001	<.0001	<.0001	0.0294
	252	121	101	252	252	252	252
LOC	0.57993	0.50610	0.61162	-0.07353	-0.07371	-0.07315	0.45218
	<.0001	<.0001	<.0001	0.2448	0.2437	0.2473	<.0001

	252	121	101	252	252	252	252
MVG	0.26107	0.53609	0.91102	-0.03999	-0.03665	-0.04028	0.18794
	<.0001	<.0001	<.0001	0.5274	0.5625	0.5244	0.0027
	252	121	101	252	252	252	252
COM	1.00000	-0.29703	-0.21127	-0.02041	-0.04296	-0.01799	0.38566
		0.0009	0.0339	0.7471	0.4972	0.7763	<.0001
	252	121	101	252	252	252	252
L_C	-0.29703	1.00000	0.77183	-0.07978	-0.06359	-0.08208	0.20122
	0.0009		<.0001	0.3844	0.4883	0.3708	0.0269
	121	121	101	121	121	121	121
M_C	-0.21127	0.77183	1.00000	-0.05784	-0.04687	-0.05787	0.02951
	0.0339	<.0001		0.5656	0.6416	0.5654	0.7695
	101	101	101	101	101	101	101
FOV	-0.02041	-0.07978	-0.05784	1.00000	0.53101	0.99915	-0.12686
	0.7471	0.3844	0.5656		<.0001	<.0001	0.0442
	252	121	101	252	252	252	252
FOC	-0.04296	-0.06359	-0.04687	0.53101	1.00000	0.54233	-0.11762
	0.4972	0.4883	0.6416	<.0001		<.0001	0.0623
	252	121	101	252	252	252	252
FOI	-0.01799	-0.08208	-0.05787	0.99915	0.54233	1.00000	-0.13087
	0.7763	0.3708	0.5654	<.0001	<.0001		0.0379
	252	121	101	252	252	252	252
FIV	0.38566	0.20122	0.02951	-0.12686	-0.11762	-0.13087	1.00000
	<.0001	0.0269	0.7695	0.0442	0.0623	0.0379	
	252	121	101	252	252	252	252
FIC	0.00351	0.12120	0.01428	-0.06023	-0.05474	-0.06001	0.45183
	0.9558	0.1854	0.8873	0.3409	0.3869	0.3428	<.0001
	252	121	101	252	252	252	252

	COM	L_C	M_C	FOV	FOC	FOI	FIV
FII	0.37333	0.18903	0.02347	-0.12610	-0.11880	-0.12981	0.98007
	<.0001	0.0378	0.8158	0.0455	0.0597	0.0395	<.0001
	252	121	101	252	252	252	252
IV	0.26675	-0.01519	-0.02997	0.12572	0.04209	0.12890	0.17537
	<.0001	0.8687	0.7661	0.0462	0.5059	0.0409	0.0052
	252	121	101	252	252	252	252
IC	0.01532	-0.03432	-0.02483	0.04519	0.16063	0.04418	0.03021
	0.8088	0.7086	0.8053	0.4751	0.0107	0.4850	0.6332
	252	121	101	252	252	252	252
II	0.28291	-0.01910	-0.03204	0.12902	0.05327	0.13524	0.17291
	<.0001	0.8352	0.7504	0.0407	0.3997	0.0319	0.0059
	252	121	101	252	252	252	252

	FIC	FII	IV	IC	II
WMC1	0.16869	0.36692	0.17182	0.03583	0.19023
	0.0073	<.0001	0.0062	0.5713	0.0024
	252	252	252	252	252
WMCV	0.31004	0.11355	0.02024	0.03252	0.03891
	<.0001	0.0720	0.7491	0.6074	0.5387
	252	252	252	252	252
DIT	0.64007	0.30014	0.00626	-0.02766	0.01525
	<.0001	<.0001	0.9212	0.6621	0.8096
	252	252	252	252	252
NOC	0.01137	-0.04719	0.11929	0.61214	0.10883
	0.8575	0.4558	0.0586	<.0001	0.0847
	252	252	252	252	252
CBO	0.09386	0.14374	0.17469	0.05131	0.18175
	0.1373	0.0225	0.0054	0.4174	0.0038
	252	252	252	252	252
	FIC	FII	IV	IC	II
LOC	0.15665	0.45173	0.13167	-0.02513	0.14509
	0.0128	<.0001	0.0367	0.6913	0.0212
	252	252	252	252	252
MVG	0.03582	0.18922	0.04752	-0.01732	0.04900
	0.5714	0.0026	0.4527	0.7844	0.4387
	252	252	252	252	252
COM	0.00351	0.37333	0.26675	0.01532	0.28291
	0.9558	<.0001	<.0001	0.8088	<.0001
	252	252	252	252	252
L_C	0.12120	0.18903	-0.01519	-0.03432	-0.01910

	0.1854	0.0378	0.8687	0.7086	0.8352
	121	121	121	121	121
M_C	0.01428	0.02347	-0.02997	-0.02483	-0.03204
	0.8873	0.8158	0.7661	0.8053	0.7504
	101	101	101	101	101
FOV	-0.06023	-0.12610	0.12572	0.04519	0.12902
	0.3409	0.0455	0.0462	0.4751	0.0407
	252	252	252	252	252
FOC	-0.05474	-0.11880	0.04209	0.16063	0.05327
	0.3869	0.0597	0.5059	0.0107	0.3997
	252	252	252	252	252
FOI	-0.06001	-0.12981	0.12890	0.04418	0.13524
	0.3428	0.0395	0.0409	0.4850	0.0319
	252	252	252	252	252
FIV	0.45183	0.98007	0.17537	0.03021	0.17291
	<.0001	<.0001	0.0052	0.6332	0.0059
	252	252	252	252	252
FIC	1.00000	0.56257	0.08842	0.08682	0.10990
		<.0001	0.1617	0.1695	0.0816
	252	252	252	252	252
FII	0.56257	1.00000	0.16846	0.02638	0.17113
	<.0001		0.0074	0.6769	0.0065
	252	252	252	252	252
	FIC	FII	IV	IC	II
IV	0.08842	0.16846	1.00000	0.14516	0.98487
	0.1617	0.0074		0.0212	<.0001
	252	252	252	252	252
IC	0.08682	0.02638	0.14516	1.00000	0.13474
	0.1695	0.6769	0.0212		0.0325
	252	252	252	252	252
II	0.10990	0.17113	0.98487	0.13474	1.00000
	0.0816	0.0065	<.0001	0.0325	
	252	252	252	252	252

Correlations between metrics for vrjuggler1

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
WMC1	1.00000	0.84969	0.07599	0.00465	0.12838	0.96132	0.89631
		<.0001	0.1502	0.9300	0.0148	<.0001	<.0001
	360	360	360	360	360	360	360
WMCV	0.84969	1.00000	0.11802	0.02326	0.16447	0.75520	0.66595
	<.0001		0.0251	0.6600	0.0017	<.0001	<.0001
	360	360	360	360	360	360	360
DIT	0.07599	0.11802	1.00000	-0.01045	0.05891	0.07671	0.06538
	0.1502	0.0251		0.8434	0.2649	0.1464	0.2159
	360	360	360	360	360	360	360
NOC	0.00465	0.02326	-0.01045	1.00000	0.16017	-0.02613	-0.03241
	0.9300	0.6600	0.8434		0.0023	0.6212	0.5399
	360	360	360	360	360	360	360
CBO	0.12838	0.16447	0.05891	0.16017	1.00000	0.14042	0.15468
	0.0148	0.0017	0.2649	0.0023		0.0076	0.0033
	360	360	360	360	360	360	360
LOC	0.96132	0.75520	0.07671	-0.02613	0.14042	1.00000	0.96477
	<.0001	<.0001	0.1464	0.6212	0.0076		<.0001
	360	360	360	360	360	360	360
MVG	0.89631	0.66595	0.06538	-0.03241	0.15468	0.96477	1.00000
	<.0001	<.0001	0.2159	0.5399	0.0033	<.0001	
	360	360	360	360	360	360	360
COM	0.91855	0.74954	0.11557	0.00607	0.19857	0.91266	0.82911
	<.0001	<.0001	0.0283	0.9086	0.0001	<.0001	<.0001
	360	360	360	360	360	360	360
L_C	-0.00979	-0.02517	-0.03669	-0.05802	-0.03606	0.01755	0.03626
	0.8943	0.7324	0.6181	0.4302	0.6242	0.8116	0.6222
	187	187	187	187	187	187	187
M_C	0.05474	0.00499	-0.03272	0.01648	-0.00820	0.12788	0.24808
	0.5087	0.9520	0.6930	0.8425	0.9212	0.1214	0.0024
	148	148	148	148	148	148	148
FOV	0.02104	0.03991	-0.11284	0.17876	0.91313	0.01198	0.02907
	0.6907	0.4503	0.0323	0.0007	<.0001	0.8207	0.5825
	360	360	360	360	360	360	360

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
FOC	-0.01973	0.02491	-0.11729	0.24650	0.60413	-0.03366	-0.03553
	0.7091	0.6375	0.0261	<.0001	<.0001	0.5244	0.5015
	360	360	360	360	360	360	360
FOI	0.01644	0.04134	-0.13367	0.16169	0.91263	0.00600	0.02258
	0.7560	0.4342	0.0111	0.0021	<.0001	0.9097	0.6693
	360	360	360	360	360	360	360
FIV	0.21109	0.26435	0.43131	0.05683	0.38628	0.23590	0.18200
	<.0001	<.0001	<.0001	0.2822	<.0001	<.0001	0.0005
	360	360	360	360	360	360	360
FIC	0.28488	0.27662	0.48897	0.00340	0.29127	0.34574	0.37676
	<.0001	<.0001	<.0001	0.9487	<.0001	<.0001	<.0001
	360	360	360	360	360	360	360
FII	0.27702	0.30925	0.44483	0.02796	0.39258	0.32995	0.32751
	<.0001	<.0001	<.0001	0.5969	<.0001	<.0001	<.0001
	360	360	360	360	360	360	360
IV	0.05428	0.04390	-0.03269	-0.00795	0.58355	0.05539	0.08325
	0.3044	0.4062	0.5364	0.8805	<.0001	0.2946	0.1148
	360	360	360	360	360	360	360
IC	0.12884	0.29141	-0.00312	0.36875	0.23207	0.09255	0.06291
	0.0144	<.0001	0.9530	<.0001	<.0001	0.0795	0.2338
	360	360	360	360	360	360	360
II	0.05764	0.05153	-0.03334	-0.00784	0.58799	0.05808	0.08536
	0.2754	0.3296	0.5284	0.8821	<.0001	0.2718	0.1059
	360	360	360	360	360	360	360

	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMC1	0.91855	-0.00979	0.05474	0.02104	-0.01973	0.01644	0.21109
	<.0001	0.8943	0.5087	0.6907	0.7091	0.7560	<.0001
	360	187	148	360	360	360	360

	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMCV	0.74954	-0.02517	0.00499	0.03991	0.02491	0.04134	0.26435
	<.0001	0.7324	0.9520	0.4503	0.6375	0.4342	<.0001
	360	187	148	360	360	360	360
DIT	0.11557	-0.03669	-0.03272	-0.11284	-0.11729	-0.13367	0.43131
	0.0283	0.6181	0.6930	0.0323	0.0261	0.0111	<.0001
	360	187	148	360	360	360	360
NOC	0.00607	-0.05802	0.01648	0.17876	0.24650	0.16169	0.05683
	0.9086	0.4302	0.8425	0.0007	<.0001	0.0021	0.2822
	360	187	148	360	360	360	360
CBO	0.19857	-0.03606	-0.00820	0.91313	0.60413	0.91263	0.38628
	0.0001	0.6242	0.9212	<.0001	<.0001	<.0001	<.0001
	360	187	148	360	360	360	360
LOC	0.91266	0.01755	0.12788	0.01198	-0.03366	0.00600	0.23590
	<.0001	0.8116	0.1214	0.8207	0.5244	0.9097	<.0001
	360	187	148	360	360	360	360
MVG	0.82911	0.03626	0.24808	0.02907	-0.03553	0.02258	0.18200
	<.0001	0.6222	0.0024	0.5825	0.5015	0.6693	0.0005
	360	187	148	360	360	360	360
COM	1.00000	-0.09114	-0.05395	0.05665	-0.03281	0.04784	0.34396
		0.2148	0.5149	0.2837	0.5349	0.3654	<.0001
	360	187	148	360	360	360	360
L_C	-0.09114	1.00000	0.78495	-0.04668	-0.03750	-0.04941	0.00229
	0.2148		<.0001	0.5258	0.6103	0.5019	0.9752
	187	187	145	187	187	187	187
M_C	-0.05395	0.78495	1.00000	-0.01026	0.00398	-0.01186	-0.08327
	0.5149	<.0001		0.9015	0.9617	0.8863	0.3143
	148	145	148	148	148	148	148
FOV	0.05665	-0.04668	-0.01026	1.00000	0.67562	0.99047	0.02916
	0.2837	0.5258	0.9015		<.0001	<.0001	0.5813
	360	187	148	360	360	360	360
FOC	-0.03281	-0.03750	0.00398	0.67562	1.00000	0.70663	-0.09692
	0.5349	0.6103	0.9617	<.0001		<.0001	0.0662
	360	187	148	360	360	360	360

	COM	L_C	M_C	FOV	FOC	FOI	FIV
FOI	0.04784	-0.04941	-0.01186	0.99047	0.70663	1.00000	0.00642
	0.3654	0.5019	0.8863	<.0001	<.0001		0.9034
	360	187	148	360	360	360	360

FIV	0.34396	0.00229	-0.08327	0.02916	-0.09692	0.00642	1.00000
	<.0001	0.9752	0.3143	0.5813	0.0662	0.9034	
	360	187	148	360	360	360	360
FIC	0.37476	0.01477	0.03602	-0.05547	-0.12201	-0.07778	0.75339
	<.0001	0.8410	0.6638	0.2939	0.0206	0.1408	<.0001
	360	187	148	360	360	360	360
FII	0.37804	0.01480	0.00492	0.00497	-0.11218	-0.01769	0.93034
	<.0001	0.8407	0.9527	0.9252	0.0333	0.7380	<.0001
	360	187	148	360	360	360	360
IV	0.11614	-0.01940	-0.00778	0.61078	-0.00569	0.59069	0.11857
	0.0276	0.7922	0.9252	<.0001	0.9143	<.0001	0.0245
	360	187	148	360	360	360	360
IC	0.14439	-0.01769	-0.03810	0.19702	0.27750	0.21701	0.09983
	0.0061	0.8101	0.6457	0.0002	<.0001	<.0001	0.0585
	360	187	148	360	360	360	360
II	0.12020	-0.01986	-0.00846	0.61367	-0.00137	0.59431	0.12119
	0.0226	0.7873	0.9187	<.0001	0.9793	<.0001	0.0214
	360	187	148	360	360	360	360

	FIC	FII	IV	IC	II
WMC1	0.28488	0.27702	0.05428	0.12884	0.05764
	<.0001	<.0001	0.3044	0.0144	0.2754
	360	360	360	360	360
WMCV	0.27662	0.30925	0.04390	0.29141	0.05153
	<.0001	<.0001	0.4062	<.0001	0.3296
	360	360	360	360	360

	FIC	FII	IV	IC	II
DIT	0.48897	0.44483	-0.03269	-0.00312	-0.03334
	<.0001	<.0001	0.5364	0.9530	0.5284
	360	360	360	360	360
NOC	0.00340	0.02796	-0.00795	0.36875	-0.00784
	0.9487	0.5969	0.8805	<.0001	0.8821
	360	360	360	360	360
CBO	0.29127	0.39258	0.58355	0.23207	0.58799
	<.0001	<.0001	<.0001	<.0001	<.0001
	360	360	360	360	360
LOC	0.34574	0.32995	0.05539	0.09255	0.05808
	<.0001	<.0001	0.2946	0.0795	0.2718
	360	360	360	360	360
MVG	0.37676	0.32751	0.08325	0.06291	0.08536
	<.0001	<.0001	0.1148	0.2338	0.1059
	360	360	360	360	360
COM	0.37476	0.37804	0.11614	0.14439	0.12020
	<.0001	<.0001	0.0276	0.0061	0.0226
	360	360	360	360	360
L_C	0.01477	0.01480	-0.01940	-0.01769	-0.01986
	0.8410	0.8407	0.7922	0.8101	0.7873
	187	187	187	187	187
M_C	0.03602	0.00492	-0.00778	-0.03810	-0.00846
	0.6638	0.9527	0.9252	0.6457	0.9187
	148	148	148	148	148
FOV	-0.05547	0.00497	0.61078	0.19702	0.61367
	0.2939	0.9252	<.0001	0.0002	<.0001
	360	360	360	360	360
FOC	-0.12201	-0.11218	-0.00569	0.27750	-0.00137
	0.0206	0.0333	0.9143	<.0001	0.9793
	360	360	360	360	360
FOI	-0.07778	-0.01769	0.59069	0.21701	0.59431
	0.1408	0.7380	<.0001	<.0001	<.0001
	360	360	360	360	360

	FIC	FII	IV	IC	II
FIV	0.75339	0.93034	0.11857	0.09983	0.12119
	<.0001	<.0001	0.0245	0.0585	0.0214
	360	360	360	360	360
FIC	1.00000	0.88739	0.03001	0.07390	0.03182
		<.0001	0.5703	0.1618	0.5473
	360	360	360	360	360
FII	0.88739	1.00000	0.09832	0.07938	0.10102
	<.0001		0.0624	0.1328	0.0555

	360	360	360	360	360	
IV	0.03001	0.09832	1.00000	0.03072	0.99978	
	0.5703	0.0624		0.5612	<.0001	
	360	360	360	360	360	
IC	0.07390	0.07938	0.03072	1.00000	0.04734	
	0.1618	0.1328	0.5612		0.3704	
	360	360	360	360	360	
II	0.03182	0.10102	0.99978	0.04734	1.00000	
	0.5473	0.0555	<.0001	0.3704		
	360	360	360	360	360	

Correlations between metric for VRJUGGLER2

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
WMC1	1.00000	0.97514	0.02089	-0.00384	0.01503	0.99876	0.99491
		<.0001	0.4020	0.8775	0.5467	<.0001	<.0001
	1611	1611	1611	1611	1611	1611	1611
WMCV	0.97514	1.00000	0.06772	-0.00550	0.03592	0.97182	0.96519
	<.0001		0.0065	0.8253	0.1496	<.0001	<.0001
	1611	1611	1611	1611	1611	1611	1611
DIT	0.02089	0.06772	1.00000	0.00050	0.16910	0.01195	0.01160
	0.4020	0.0065		0.9839	<.0001	0.6318	0.6418
	1611	1611	1611	1611	1611	1611	1611
NOC	-0.00384	-0.00550	0.00050	1.00000	0.31419	-0.00401	-0.00410
	0.8775	0.8253	0.9839		<.0001	0.8721	0.8693
	1611	1611	1611	1611	1611	1611	1611
CBO	0.01503	0.03592	0.16910	0.31419	1.00000	0.00925	0.01369
	0.5467	0.1496	<.0001	<.0001		0.7106	0.5830
	1611	1611	1611	1611	1611	1611	1611
LOC	0.99876	0.97182	0.01195	-0.00401	0.00925	1.00000	0.99648
	<.0001	<.0001	0.6318	0.8721	0.7106		<.0001
	1611	1611	1611	1611	1611	1611	1611
MVG	0.99491	0.96519	0.01160	-0.00410	0.01369	0.99648	1.00000
	<.0001	<.0001	0.6418	0.8693	0.5830	<.0001	
	1611	1611	1611	1611	1611	1611	1611

	COM	L_C	M_C	FOV	FOC	FOI	FIV
WMC1	0.99503	-0.01638	-0.01798	-0.00960	-0.01176	-0.00957	0.05253
	<.0001	0.7257	0.7553	0.7001	0.6373	0.7010	0.0350
	1611	461	303	1611	1611	1611	1611
WMCV	0.96740	-0.00181	0.00263	-0.01809	-0.01917	-0.01799	0.11784
	<.0001	0.9692	0.9636	0.4680	0.4420	0.4706	<.0001
	1611	461	303	1611	1611	1611	1611
DIT	0.01881	-0.05311	-0.06715	-0.05948	-0.06992	-0.06190	0.52132
	0.4506	0.2551	0.2439	0.0170	0.0050	0.0130	<.0001
	1611	461	303	1611	1611	1611	1611
NOC	-0.00400	-0.01609	-0.04195	0.34707	0.44736	0.34307	-0.00907
	0.8724	0.7304	0.4670	<.0001	<.0001	<.0001	0.7160
	1611	461	303	1611	1611	1611	1611
CBO	0.01171	-0.08313	-0.11861	0.90258	0.84923	0.90286	0.37137
	0.6385	0.0746	0.0391	<.0001	<.0001	<.0001	<.0001
	1611	461	303	1611	1611	1611	1611
LOC	0.99526	-0.01742	-0.02164	-0.00933	-0.01078	-0.00930	0.03809
	<.0001	0.7092	0.7075	0.7082	0.6656	0.7091	0.1265
	1611	461	303	1611	1611	1611	1611
MVG	0.99252	-0.01945	-0.01498	-0.01014	-0.01180	-0.00995	0.04445
	<.0001	0.6770	0.7952	0.6841	0.6359	0.6898	0.0745
	1611	461	303	1611	1611	1611	1611

	FIC	FII	IV	IC	II	
WMC1	0.05208	0.05553	0.00654	0.00022	0.00679	
	0.0366	0.0258	0.7931	0.9928	0.7855	
	1611	1611	1611	1611	1611	
WMCV	0.13099	0.12218	-0.00188	-0.00148	-0.00149	
	<.0001	<.0001	0.9401	0.9528	0.9522	
	1611	1611	1611	1611	1611	
DIT	0.52569	0.52608	0.10193	0.06212	0.10204	
	<.0001	<.0001	<.0001	0.0126	<.0001	
	1611	1611	1611	1611	1611	
NOC	-0.01386	-0.01016	0.12946	0.34026	0.12942	
	0.5783	0.6838	<.0001	<.0001	<.0001	
	1611	1611	1611	1611	1611	
CBO	0.35312	0.37530	0.18025	0.13189	0.18081	
	<.0001	<.0001	<.0001	<.0001	<.0001	

	1611	1611	1611	1611	1611	
LOC	0.03697	0.04153	0.00028	-0.00091	0.00056	
	0.1381	0.0956	0.9911	0.9708	0.9821	
	1611	1611	1611	1611	1611	
MVG	0.04345	0.05324	0.00008	-0.00055	0.00067	
	0.0813	0.0326	0.9974	0.9825	0.9786	
	1611	1611	1611	1611	1611	

	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
COM	0.99503	0.96740	0.01881	-0.00400	0.01171	0.99526	0.99252
	<.0001	<.0001	0.4506	0.8724	0.6385	<.0001	<.0001
	1611	1611	1611	1611	1611	1611	1611
L_C	-0.01638	-0.00181	-0.05311	-0.01609	-0.08313	-0.01742	-0.01945
	0.7257	0.9692	0.2551	0.7304	0.0746	0.7092	0.6770
	461	461	461	461	461	461	461
M_C	-0.01798	0.00263	-0.06715	-0.04195	-0.11861	-0.02164	-0.01498
	0.7553	0.9636	0.2439	0.4670	0.0391	0.7075	0.7952
	303	303	303	303	303	303	303
FOV	-0.00960	-0.01809	-0.05948	0.34707	0.90258	-0.00933	-0.01014
	0.7001	0.4680	0.0170	<.0001	<.0001	0.7082	0.6841
	1611	1611	1611	1611	1611	1611	1611
FOC	-0.01176	-0.01917	-0.06992	0.44736	0.84923	-0.01078	-0.01180
	0.6373	0.4420	0.0050	<.0001	<.0001	0.6656	0.6359
	1611	1611	1611	1611	1611	1611	1611
FOI	-0.00957	-0.01799	-0.06190	0.34307	0.90286	-0.00930	-0.00995
	0.7010	0.4706	0.0130	<.0001	<.0001	0.7091	0.6898
	1611	1611	1611	1611	1611	1611	1611
FIV	0.05253	0.11784	0.52132	-0.00907	0.37137	0.03809	0.04445
	0.0350	<.0001	<.0001	0.7160	<.0001	0.1265	0.0745
	1611	1611	1611	1611	1611	1611	1611

	COM	L_C	M_C	FOV	FOC	FOI	FIV
COM	1.00000	-0.03889	-0.03888	-0.00996	-0.01160	-0.00978	0.04488
		0.4048	0.5001	0.6896	0.6416	0.6949	0.0717
	1611	461	303	1611	1611	1611	1611
L_C	-0.03889	1.00000	0.88666	-0.06624	-0.04603	-0.06811	-0.04552
	0.4048		<.0001	0.1556	0.3241	0.1443	0.3294
	461	461	303	461	461	461	461
M_C	-0.03888	0.88666	1.00000	-0.06912	-0.04737	-0.06678	-0.08707
	0.5001	<.0001		0.2303	0.4113	0.2465	0.1305
	303	303	303	303	303	303	303
FOV	-0.00996	-0.06624	-0.06912	1.00000	0.94881	0.99893	-0.05697
	0.6896	0.1556	0.2303		<.0001	<.0001	0.0222
	1611	461	303	1611	1611	1611	1611
FOC	-0.01160	-0.04603	-0.04737	0.94881	1.00000	0.95277	-0.08127
	0.6416	0.3241	0.4113	<.0001		<.0001	0.0011
	1611	461	303	1611	1611	1611	1611
FOI	-0.00978	-0.06811	-0.06678	0.99893	0.95277	1.00000	-0.05874
	0.6949	0.1443	0.2465	<.0001	<.0001		0.0184
	1611	461	303	1611	1611	1611	1611
FIV	0.04488	-0.04552	-0.08707	-0.05697	-0.08127	-0.05874	1.00000
	0.0717	0.3294	0.1305	0.0222	0.0011	0.0184	
	1611	461	303	1611	1611	1611	1611

	FIC	FII	IV	IC	II
COM	0.03682	0.04828	0.00025	-0.00091	0.00054
	0.1396	0.0527	0.9920	0.9708	0.9828
	1611	1611	1611	1611	1611
L_C	-0.00507	-0.05311	-0.02894	-0.00507	-0.02931
	0.9135	0.2551	0.5354	0.9135	0.5302
	461	461	461	461	461
M_C	-0.05671	-0.09370	-0.02276	-0.04011	-0.02312
	0.3252	0.1035	0.6931	0.4866	0.6885
	303	303	303	303	303
FOV	-0.05920	-0.05802	0.14184	0.12316	0.14187
	0.0175	0.0199	<.0001	<.0001	<.0001
	1611	1611	1611	1611	1611
FOC	-0.07312	-0.08235	0.05100	0.15479	0.05096
	0.0033	0.0009	0.0407	<.0001	0.0408
	1611	1611	1611	1611	1611
FOI	-0.06041	-0.05966	0.13998	0.12172	0.14004

		0.0153	0.0166	<.0001	<.0001	<.0001	
		1611	1611	1611	1611	1611	
	FIV	0.93960	0.98891	0.11955	0.04472	0.11991	
		<.0001	<.0001	<.0001	0.0728	<.0001	
		1611	1611	1611	1611	1611	
	WMC1	WMCV	DIT	NOC	CBO	LOC	MVG
FIC	0.05208	0.13099	0.52569	-0.01386	0.35312	0.03697	0.04345
	0.0366	<.0001	<.0001	0.5783	<.0001	0.1381	0.0813
	1611	1611	1611	1611	1611	1611	1611
FII	0.05553	0.12218	0.52608	-0.01016	0.37530	0.04153	0.05324
	0.0258	<.0001	<.0001	0.6838	<.0001	0.0956	0.0326
	1611	1611	1611	1611	1611	1611	1611
IV	0.00654	-0.00188	0.10193	0.12946	0.18025	0.00028	0.00008
	0.7931	0.9401	<.0001	<.0001	<.0001	0.9911	0.9974
	1611	1611	1611	1611	1611	1611	1611
IC	0.00022	-0.00148	0.06212	0.34026	0.13189	-0.00091	-0.00055
	0.9928	0.9528	0.0126	<.0001	<.0001	0.9708	0.9825
	1611	1611	1611	1611	1611	1611	1611
II	0.00679	-0.00149	0.10204	0.12942	0.18081	0.00056	0.00067
	0.7855	0.9522	<.0001	<.0001	<.0001	0.9821	0.9786
	1611	1611	1611	1611	1611	1611	1611
	COM	L_C	M_C	FOV	FOC	FOI	FIV
FIC	0.03682	-0.00507	-0.05671	-0.05920	-0.07312	-0.06041	0.93960
	0.1396	0.9135	0.3252	0.0175	0.0033	0.0153	<.0001
	1611	461	303	1611	1611	1611	1611
FII	0.04828	-0.05311	-0.09370	-0.05802	-0.08235	-0.05966	0.98891
	0.0527	0.2551	0.1035	0.0199	0.0009	0.0166	<.0001
	1611	461	303	1611	1611	1611	1611
IV	0.00025	-0.02894	-0.02276	0.14184	0.05100	0.13998	0.11955
	0.9920	0.5354	0.6931	<.0001	0.0407	<.0001	<.0001
	1611	461	303	1611	1611	1611	1611
IC	-0.00091	-0.00507	-0.04011	0.12316	0.15479	0.12172	0.04472
	0.9708	0.9135	0.4866	<.0001	<.0001	<.0001	0.0728
	1611	461	303	1611	1611	1611	1611
II	0.00054	-0.02931	-0.02312	0.14187	0.05096	0.14004	0.11991
	0.9828	0.5302	0.6885	<.0001	0.0408	<.0001	<.0001
	1611	461	303	1611	1611	1611	1611
		FIC	FII	IV	IC	II	
	FIC	1.00000	0.95012	0.06197	0.02914	0.06302	
			<.0001	0.0129	0.2424	0.0114	
		1611	1611	1611	1611	1611	
	FII	0.95012	1.00000	0.11673	0.04382	0.11789	
		<.0001		<.0001	0.0787	<.0001	
		1611	1611	1611	1611	1611	
	IV	0.06197	0.11673	1.00000	0.38875	0.99996	
		0.0129	<.0001		<.0001	<.0001	
		1611	1611	1611	1611	1611	
	IC	0.02914	0.04382	0.38875	1.00000	0.38878	
		0.2424	0.0787	<.0001		<.0001	
		1611	1611	1611	1611	1611	
	II	0.06302	0.11789	0.99996	0.38878	1.00000	
		0.0114	<.0001	<.0001	<.0001		
		1611	1611	1611	1611	1611	

Vita:

Andrew Ray came to the discipline of software engineering through the off chance that he enjoyed using computers in high school. During high school his parents thought that he would strike it rich by going to the outside world so they said "Andrew, why don't you move away from there." So off to the bright lights of Virginia Tech he went. Several years later, one of the chapters in the tale of the country boy who went to the city to get an education is nearing completion.

Andrew holds a Bachelor of Science degree in Computer Science from Virginia Tech where he studied a wide variety of subjects both in and outside of Computer Science. He has worked at the VT CAVE, IBM, VBI, and has been the systems administrator for the Mid-Atlantic programming contest for the past three years. The next major step in his professional career is pursuing a Ph.D.

I would like to thank God for allowing me to make it this far. Without him, I would be nothing. I would like to thank my parents for bringing me into this world and providing care and love to me till this day. Without them I wouldn't have had the drive to attend college. I would like to thank Dr. Henry for all of her guidance and help with my education. Without her guidance, I probably would have not gone to graduate school.