A Quality Comparison of Kauli Community Software for Higher Education

Charles W. Butler
Colorado State University

Abstract: The Kauli Foundation provides open source community software for universities. The foundation is comprised of many universities, colleges, and commercial firms who have joined together to develop an application portfolio for higher education administration. This research studied the software quality of three Kauli applications; specifically, Kauli Financial System, Kauli Rice and Kauli Coeus. The analysis of software quality utilized McCabe metrics as a reference base. McCabe metrics include software metrics, cyclomatic complexity, essential complexity, and method design complexity, for determining software quality and reliability.

INTRODUCTION

In the 2012, the growing market of open community software offered through the Kuali Foundation was explored. (Butler, 2012) Indiana University is the founding partner in Kuali. Today, a growing community of 77 universities, colleges and commercial firms have joined together to form the Kuali Foundation Community Members. These organizations are building and sustaining open-source software for higher education, by higher education. In addition, the 2012 article presented the results of analyzing Kuali software quality using contemporary software metrics. The results of that analysis provided an indication of whether the outcome of the consortium development strategy is one that is risky for a university. This paper takes the analysis of Kauli software quality to an expanded level. In this paper, the results are presented from analyzing additional Kauli software using contemporary software metrics. The results of this analysis are presented to provide an expanded view of whether the outcome of the consortium development strategy is one that is not risky or risky for a university.

KUALI FOUNDATION

As previously written, the Kuali Foundation is a consortium of interested universities, colleges and commercial firms that have joined together to produce an enterprise software solution for the academic business model. The Kauli Foundation utilizes a community source developed software acquisition methodology. TABLE 1 contains a number of the Carnegie Mellon class universities participating in the foundation. In addition, other leading commercial firms such as IBM and VMware are participants. The consortium pools resources to develop and sustain many of the software systems needed for higher education. This approach reduces costs and produces software that better fits institutional needs. The Kuali Foundation is funded through a fee-based membership starting at $4,500 growing to $24,500 based upon university budget size. These universities share information technology (IT) resources to develop software and the software can be used by anyone without a purchase or maintenance fee. There are 15

lead universities who share IT resources and coordinate project activity as numerous "virtual" projects. These institutions freely share work with the world as that fits the public service mission of colleges and universities. Working collaboratively together, the outcome is shared best practices and reduced costs beyond legacy approaches to purchased software. Since the 2012 article was published, the Kuali membership has grown from 67 to 77 international research universities, community colleges, public and private institutions, and commercial affiliates. (Kuali Foundation, 2014)

TABLE 1
CARNEGIE MELLON CLASS KAULI UNIVERSITY MEMBERS

Boston College	Michigan State University
Boston University	University of Arizona
Clemson University	University of Arkansas
Colorado State University	University of California - Berkeley
Cornell University	University of Florida
Clemson University	University of Hawaii
Cornell University	University of Illinois
Indiana University	University of Michigan

Kuali Software

The Kuali software is a portfolio of applications ranging from financial to mobile connectivity. Below is a short summary of each application: (Kuali Foundation, 2014)

- Kauli Financial System (KFS) (2005): Financial software that meets the needs of Carnegie Class institutions; current Version 5.0.2
- Kauli Coeus (2006): Research administration for grants administration to federal funding agencies; current Version 5.2
- Kauli Rice (2007): Middleware products that integrate products allowing applications to be built in an agile fashion; current Version 2.3.3
- Kauli Student (KS) (2007): Business needs of students, faculty and institutions throughout the academic lifecycle; current Version 2.0.2
- Kauli Open Library Environment (2010): Integration of academic and research libraries for managing and delivering intellectual information; current Version 1.0
- Kauli Mobility (2011): Connect mobile devices to a variety of campus systems; current Version 2.4
- Kauli People for the Enterprise: HR/Payroll System built by higher education for higher education; current Version 2.0.1
- Ready (scheduled 2014): A business continuity planning tool

These applications are in various stages of development and implementation. Currently, all of the consortium universities and colleges are using or are in deployment phases of the Financial System.

QUALITY SOFTWARE METRICS

Analyzing software quality can be subjective or objective methodology. As a basis for software quality, NASA has a reference model given the agency's mission critical, high performance standards. The following three metrics are used in NASA's Metrics Data Program (MDP) repository for determining software quality and reliability. (NASA, n.d., p. 4)

- Halstead Metrics: Halstead's metrics assumes that a program should be viewed as an expression of language. Halstead expressed mathematically the relationships among the number of variables, the type of programming statements and the complexity of the code. Unfortunately, Halstead

metrics are difficult to compute. In order to be useful, a metric must be computed quickly and easily understood. (Kaur, Minhas, Mehan, & Kakkar, 2009)

- McCabe Cyclomatic Complexity: McCabe's cyclomatic complexity measures the number of linearly-independent paths through a program method. The basic assumption is that software complexity is directly related to the number of control paths generated by the code. This metric is easy to understand. In fact, this metric has been criticized because it seems too simple. However, the McCabe metric is an easy-to-compute, high-level measure of a program's complexity which has been shown to agree with empirical data. (Kaur, Minhas, Mehan, & Kakkar, 2009)
- Lines of code metrics: The lines of code metric seems easy to measure. While a longer program can be more prone to error than a short program, that is not always true. In addition, newer object oriented coding techniques make it difficult to determine the actual lines of code executed.

Reviewing the competing metrics, McCabe's cyclomatic complexity is an appropriate metric to predict software reliability. It has been validated by two NASA studies. (NASA IV&V Facility, 2008). It might also aid in projecting the cost of software support. There are several applications available to measure cyclomatic complexity. However, McCabe IQ is an automated tool that has an array of graphical presentations and is widely accepted by software developers. Therefore, McCabe IQ was chosen as the automated tool for this study. Based upon prior research results, the following McCabe metrics were determined for Kuali applications:

- Cyclomatic complexity: a measurement of the size of a software method's decision logic. Cyclomatic complexity, v, is determined for each software program's methods. Research has shown that when the cyclomatic complexity of a method exceeds 10, then its reliability degrades exponentially. So, a $v > 10$ is considered low quality and riskier software. (McCabe, 1976)
- Essential complexity: a measurement of a software method's decision structure or architecture. Essential complexity, ev, is determined for each software program's methods. Research shows that when essential complexity of a method grows higher than 1, it is harder to maintain. So, software with an $ev > 3$ is considered to be harder and more difficult to maintain than software with $ev = 1$. (McCabe, 1976)
- Module design complexity: a measurement of a software method's integration decision structure with other methods. Module design complexity, iv, is determined for each software program's methods. Research shows that when module design complexity of a method grows high, the level of integration testing is increased. More integration results in higher operational risk. So, when iv is a high proportion of a method's decision structure, it is an integration risk. (McCabe & Butler, 1989)
- Design complexity: a measurement of the decision structure which controls the invocation of modules within the design. Design complexity, S_0, is a quantification of the testing effort of the calls in the design, starting with the top or root module, trickling down through subordinates and exiting through the top. Design complexity is calculated as $\sum iv$. (McCabe & Butler, 1989)
- Integration complexity: a measurement of the integration tests that qualify the design tree. Integration complexity, S_1, is a quantification of a basis set of integration tests and measures the minimum integration testing effort of a design. Each S_1 test validates the integration of several modules and is known as a subtree of the whole design tree. Integration complexity is calculated as $S_1 = S_0 - n + 1$ where n = number of modules in the design. (McCabe & Butler, 1989)

KAULI FINANCIAL SYSTEM APPLICATION ANALYSIS

The first application studied was the Kauli Financial System. The Kuali Financial System project is working to create and enhance a comprehensive suite of financial software that meets the needs of Carnegie Class institutions. The financial system consists of 10 modules identified as follows:

1. Accounts Receivable (AR)
2. Budget Construction (BC)
3. Capital Asset Builder (CAB)
4. Capital Asset Management (CAM)
5. Contract and Grants (CG)
6. Effort Certification (EC)
7. Endowment Management (EM)
8. External (EX)
9. Labor Distribution (LD)
10. Purchasing/Accounts Payable (PUR/AP)

For the purposes of this study, these ten modules were analyzed using McCabe IQ. The McCabe parsing was completed on Kauli Financial System, Version 4.1.1.

The Kauli Financial System application profile is summarized in TABLE 2. The profile includes averages calculated based on the number of parsed methods. The relatively low averages are a result of the large number of methods that fall below the McCabe cyclomatic complexity threshold of 10. (McCabe Staff) In this case, the term threshold means the value of a metric above which there is an element of interest, specifically about quality risks. Typically, below a threshold, metrics do not correlate with high work effect, and code elements below a threshold usually do not require code to be reviewed or modified. (McCabe Staff, n.d.)

TABLE 2
KAULI FINANCIAL SYSTEM APPLICATION PROFILE

	Module Name									
Static Metric	AR	BC	CAB	CAM	CG	EC	EM	EX	LD	PUR/AP
Number of Methods	3027	6183	874	2226	702	663	4536	433	2069	5918
Total SLOC	17753	31543	5725	12011	2649	3535	28099	1984	13365	37935
S_0	5336	8431	1797	3833	940	1030	8011	673	3600	11463
S_1	2310	2249	924	1608	239	368	3476	241	1532	5546
Total v	5565	8,669	1914	4012	968	1077	8506	721	3777	12150
Average v	1.84	1.40	2.19	1.80	1.36	1.62	1.88	1.67	1.83	2.05
Maximum v	65	39	21	39	17	12	26	30	64	84
Total ev	3856	6874	1234	2787	764	774	5750	518	2692	8118
Average ev	1.27	1.11	1.41	1.24	1.09	1.17	1.27	1.20	1.30	1.37
Maximum ev	65	30	13	18	43	12	26	21	54	43
Average iv	1.76	1.36	2.06	1.72	1.34	1.55	1.77	1.55	1.72	1.94
Maximum iv	65	32	17	38	13	11	49	22	60	82

As seen in TABLE 2, the Kauli Financial System exhibits positive software metrics. The total parsed source lines of code is 191,994. As highlighted in orange, Budget Construction, Endowment Management, and Purchasing/Accounts Payable are the three largest modules. Purchasing/Accounts Payable is the largest module (S_0=11,463), and it requires the most high level integration testing (S_1=5546). Parsing by McCabe IQ resulted in 26,631 methods with the following profile metrics:

- The average cyclomatic complexity ranged from 1.36 to 2.19.
- The average essential complexity ranged from 1.09 to 1.41.
- The average method design complexity ranged from 1.34 to 2.06.
- The total executable lines of code per module ranged from 2,649 to 37,935.

As indicated in green in Table 2, Contract and Grants is the highest quality module with the lowest average v = 1.36 and is a well-structured method with the best maintainability characteristics given an average ev = 1.09. Budget Construction has similar low cyclomatic and essential complexities with an average v = 1.40 and ev = 1.11. Capital Asset Builder (highlighted in yellow) is the lowest quality module with average v = 2.19 and average ev = 1.41. External Funds required the lowest amount of unit level testing using McCabe Structure Testing Methodology, since its total v = 721. Contracts and Grants is also the module that requires the least amount of high level integration testing because it has the lowest S_1 = 239.

The Kauli Financial System risk profile in TABLE 3 contains the metrics for the high risk methods for the application (methods with a v > 10). This table accentuates existing low quality within KFS code. The total parsed source lines of code for high risk methods is 15,942 (8.3% of the total source lines of code). For KFS, 375 methods (1.41% of the total number of methods) are considered high risk. Its average cyclomatic, essential, and module design complexities are much higher than overall profile metrics. Based on the high risk profile, one can conclude which modules are riskiest, poorly structured, and require the highest integration effort.

TABLE 3
KAULI FINANCIAL SYSTEM RISK PROFILE (v(G)>10)

	Module Name									
Static Metric	AR	BC	CAB	CAM	CG	EC	EM	EX	LD	PUR/AP
Total methods	3027	6183	874	2226	702	663	4536	433	2069	5918
High risk methods	44	45	13	29	1	1	76	3	37	126
% high risk of total methods	1.50%	0.72%	1.48%	1.30%	0.14%	0.15%	0.72%	1.68%	1.78%	2.12%
Total SLOC	17753	31543	5725	12011	2649	3535	28099	1984	13365	37935
SLOC	3140	2750	491	1532	58	43	4039	189	2594	7982
Total v	790	676	175	457	17	12	1173	63	718	2243
Average v	17.95	15.02	13.46	15.76	17.00	12.00	15.43	21.00	19.41	17.94
Max v	65	39	21	39	17	12	49	30	64	84
Total ev	353	267	104	156	11	12	636	40	332	994
Average ev	8.02	5.93	8.00	5.38	11.00	12.00	8.37	7.51	8.97	7.95
Max ev	65	30	13	18	11	12	26	21	54	43
Total iv	736	612	151	398	13	11	1012	49	667	2027
Average iv	16.73	13.60	11.62	13.72	13	11	13.32	16.33	18.03	16.22
Max iv	65	32	17	38	13	11	49	22	60	82

Contract and Grants, Effort Certification, and External Funds have only 1, 1, and 3 high risk methods, respectively, and are not considered in the following analysis. The remaining key profile metrics were calculated:

- The average cyclomatic complexity ranged from 13.46 to 19.41 with an individual method high of 84.
- The average essential complexity ranged from 5.38 to 8.97 with an individual method high of 65.
- The average module design complexity ranged from 11.62 to 18.03.
- The total executable lines of code per module ranged from 491 to 7,982.

The high risk profile for KFS indicates that KFS has low quality, high risk methods. One module, Purchasing Accounts Payable (highlighted in red), has a percentage of high risk methods to total methods greater than 2%. Labor Distribution has the highest average v, ev, and iv at 19.41, 8.97 and 18.03, respectively and has the riskiest, high risk methods. Purchasing/Accounts Payable has 126 high risk methods whose average v is 17.94. Labor Distribution (also highlighted in red) has the lowest quality, high risk methods with average v, ev, and iv of 19.41, 8.97, and 18.03, respectively. Capital Asset Management and Budget Construction are the best structured modules with the highest maintainability characteristics with average ev = 5.38 and 5.93, respectively.

KAULI RICE APPLICATION ANALYSIS

The second application studied was the Kauli Rice. The Kuali Rice software provides an enterprise class middleware suite of integrated products that allows for applications to be built in an agile fashion. This enables developers to react to end-user business requirements in an efficient and productive manner, so that they can produce high quality business applications. Kauli Rice consists of seven modules identified as follows:

1. Kauli Enterprise Notification (KEN)
2. Kauli Enterprise Workflow (KEW)
3. Kauli Identity Management (KIM)
4. Kauli Nervous System (KNS)
5. Kauli Rapid Development (KRD)
6. Kauli Rules Management (KRM)
7. Kauli Service Bus (KSB)

For the purposes of this study, these seven modules were analyzed using McCabe IQ. The McCabe parsing was completed on Kauli Rice, Version 2.1.1.

The Kauli Rice application profile is summarized in TABLE 4. The profile includes averages calculated based on the number of parsed methods. As seen in the table, Rice exhibits positive software metrics. Rapid Development (orange in Table 4) is the largest module (S_0=8453), and it requires the most high level integration testing (S_1=3624).

TABLE 4
KAULI RICE APPLICATION PROFILE

Static Metrics	Module Names						
	KEN	KEW	KIM	KNS	KRD	KRM	KSB
Number of Methods	351	2108	2707	2617	4830	1621	1248
Total SLOC	1024	6999	13746	18274	26409	9035	6076
S_0	400	2574	4519	5782	8453	2843	1967
S_1	50	467	1813	3166	3624	1223	720
Total v	400	2610	4711	6205	9117	2974	2093
Average v	1.14	1.24	1.74	2.37	1.89	1.83	1.68
Max v	10	11	47	83	55	16	22
Total ev	351	2183	3351	3747	6185	2074	1488
Average ev	1	1.04	1.24	1.43	1.28	1.28	1.19
Max ev	1	7	28	23	54	13	13
Average iv	1.14	1.22	1.67	2.21	1.75	1.75	1.58
Max iv	10	11	45	80	37	14	18

The total parsed source lines of code is 191,994. Parsing by McCabe IQ resulted in 15,482 methods with the following profile metrics:

- The average cyclomatic complexity ranged from 1.14 to 2.37.
- The average essential complexity ranged from 1.00 to 1.43.
- The average module design complexity ranged from 1.14 to 2.21.
- The total executable lines of code per module ranged from 1,024 to 26,409.

Using McCabe software metrics as the criteria, Enterprise Notification (highlighted in green) is the highest quality module with an average v = 1.14 and it is also the best structured module with an average ev = 1.00 which indicates that it contains perfectly structured code. Enterprise Workflow is well-structured with an average ev = 1.04. Nervous System (also highlighted in yellow) is the lowest quality module with an average v = 2.37 and average ev = 1.43. Overall, the Rice modules are well-written with methods that are maintainable, since the average ev for all the modules is 1.25. Enterprise Notification required the lowest amount of unit level testing using McCabe Structure Testing Methodology, since its total v = 400, and it is also the module that requires the least amount of high level integration testing because it has the lowest S_1 = 50. In fact, Enterprise Notification is a good example of size matters. It is the smallest module in terms of lines of code and it exhibits high quality measurements for each software metric.

The Rice risk profile in TABLE 5 displays the metrics for the high risk methods for this application (methods with a v > 10). This table accentuates existing low quality within Rice code. The total parsed source lines of code for high risk methods is 15942 (8.3% of the total source lines of code). For Rice, 375 methods (1.41% of the total number of methods) are considered high risk. Its average cyclomatic, essential, and method design complexities are much higher than overall profile metrics. Based on the high risk profile, one can conclude which methods are riskiest, poorly structured, and require the highest integration effort.

TABLE 5
KAULI RICE RISK PROFILE (v(G)>10)

Static Metric	KEN	KEW	KIM	KNS	KRD	KRM	KSB
Total methods	351	2108	2707	2617	4830	1621	1248
High risk methods	0	1	24	84	90	11	5
% high risk of total methods	0	0.00%	0.88%	1.30%	3.21%	0.68%	0.40%
Total SLOC	1024	6999	13746	18274	26409	9035	6076
SLOC	0	11	1462	4779	4477	415	225
Total v	0	11	476	1471	1535	139	82
Average v	0.00	11	19.83	17.51	17.06	12.64	16.40
Max v	0	11	47	83	55	16	22
Total ev	0	1	208	636	695	109	45
Average ev	0	1	8.67	7.57	7.72	9.91	9.00
Max ev	0	1	28	23	54	13	13
Total iv	0	11	455	1336	1301	120	72
Average iv	0	11	18.54	15.9	14.46	10.91	14.40
Max iv	0	1	45	80	37	14	18

Enterprise Notification and Enterprise Workflow have zero and one high risk methods, respectively, and are not used in the below analysis. For the remaining high risk methods, the following key profile metrics were calculated:

- The average cyclomatic complexity ranged from 12.64 to 19.83 with an individual method high of 83.
- The average essential complexity ranged from 7.57 to 9.91 with an individual method high of 54.
- The average module design complexity ranged from 10.91 to 18.54.
- The total executable lines of code per module ranged from 225 to 4,779.

The high risk profile for Rice indicates that Rice has low quality, high risk methods. One module, Rapid Development (shown as red), has a percentage of high risk methods to total methods greater than 2% at 3.21%. Identity Management has the highest average v = 19.83. Rapid Development has 90 high risk methods whose average v = 17.06. Identify Management has the highest average ev = 8.67 which indicates that it unstructured and less maintainable. Both Nervous System and Rapid Development have high ev averages, and they have many high risk methods. Identify Management high risk methods exhibit the highest level of method integration with an average iv = 18.54. Overall, Rule Management (highlighted in yellow) has the highest quality, high risk methods.

KAULI COEUS APPLICATION ANALYSIS

The third application studied was the Kauli Coeus. Kuali Coeus for Research Administration is a comprehensive system to manage the complexities of research administration needs from the faculty researcher through grants administration to federal funding agencies. Kauli Coeus consists of 10 modules identified as follows:

1. Award (AW)
2. Budget (BU)
3. Conflict of Interest (CI)
4. IACUS (IA)
5. Institutional Proposal (IP)
6. Institutional Review Board (IRB) Human Rights (IR)
7. Negotiations (NE)
8. Proposal and Budget Development (PBD)
9. Grants.gov S2S Submission (S2S)
10. Subawards (SU)

For the purposes of this study, these ten modules were analyzed using McCabe IQ. The McCabe parsing was completed on Kauli Coeus, Version 5.0.1.

The Kauli Coeus application profile is summarized in TABLE 6. The profile includes averages calculated based on the number of parsed methods. As seen in the table, Coeus exhibits positive software metrics but there are three relative large modules, Award, Proposal and Budget Development, and Grants.gov 2S2 Submission (all highlighted in orange). Award is the largest module with an S_0 = 8525. The total parsed source lines of code is 165,105.

TABLE 6
KAULI COEUS APPLICATION PROFILE

Static Metric	Module Name									
	AW	BU	CI	IA	IP	IR	NE	PBD	S2S	SU
Number of Methods	4758	2613	2078	2958	1645	5262	639	3323	1881	662
Total SLOC	27087	18510	9558	14154	8103	26535	3317	21313	33351	3177
S_0	8525	5429	3261	4505	2638	8718	1079	6333	7348	983
S_1	3758	2817	1184	1548	994	3457	441	3011	5488	322
Total v	9281	5850	3416	4789	2823	9187	1133	6746	7912	1050
Average v	1.95	2.24	1.64	1.62	1.72	1.75	1.77	2.02	4.25	1.59
Max v	57	97	23	21	34	42	29	43	118	37
Total ev	6221	3867	2491	3522	2014	6498	790	4404	3388	811
Average ev	1.30	1.48	1.20	1.19	1.22	1.23	1.24	1.33	1.82	1.23
Max ev	44	97	16	15	34	29	23	43	59	36
Average iv	1.79	2.08	1.57	1.52	1.60	1.66	1.69	1.91	3.95	1.48
Max iv	57	33	23	21	27	42	27	33	105	27

Parsing by McCabe IQ resulted in 25,819 methods with the following profile metrics:

- The average cyclomatic complexity ranged from 1.59 to 4.25.
- The average essential complexity ranged from 1.19 to 1.82.
- The average module design complexity ranged from 1.48 to 3.95.
- The total executable lines of code per module ranged from 3,177 to 27,087.

The averages and upper limits of v, ev, and iv are higher than the Financial System and Rice. Subawards (colored green in Table 6) is the highest quality module with the lowest average v = 1.59, a relative low average ev = 1.23, and the lowest average iv = 1.48. Grants.gov S2S Submission is lowest quality module with the highest average v = 4.25, the highest average ev = 1.82, and the highest iv = 3.95.

The Coeus risk profile in TABLE 7 contains information about low quality methods (methods with v > 10). The total parsed source lines of code for high risk methods is 27,799 (16.8% of the total source lines of code). For Coeus, 511 methods (2.00% of the total number of methods) are considered high risk. Its high risk average cyclomatic, essential, and module design complexities are much higher than overall profile metrics.

For the evaluated high risk methods, the following key profile metrics were calculated:

- The average cyclomatic complexity ranged from 14.80 to 22.83 with an individual method high of 97.
- The average essential complexity ranged from 5.76 to 14.50 with an individual method high of 97.
- The average module design complexity ranged from 10.85 to 18.33.
- The total executable lines of code per module ranged from 405 to 19,108.

Highlighted in red, Budget and Grants.gov S2S Submission have the highest % of high risk to total methods greater than 2% at 2.37% and 11.06%, respectively. Coeus modules have an average v = 17.94, which is high and risky. Five modules have average ev > 9.00, and the application has high method integration complexity with four module average iv > 15. Based on the high risk profile, one can conclude that Coeus high risk modules have lower quality than the Financial System and Rice applications.

TABLE 7
KAULI COEUS RISK PROFILE (v(G)>10)

Static Metric	Module Name									
	AW	BU	CI	IA	IP	IR	NE	PBD	S2S	SU
Total methods	4758	2613	2078	2958	1645	5262	639	3323	1881	662
High risk methods	65	62	18	29	18	42	9	54	208	6
% high risk of total methods	1.37%	2.37%	0.86%	0.98%	1.09%	0.80%	1.41%	1.63%	11.06%	0.91%
Total SLOC	27087	18510	9558	14154	8103	26535	3317	21313	33351	3177
SLOC	3102	3309	691	727	849	1673	457	2663	19108	405
Total v	1107	1178	271	296	290	626	149	860	4120	137
Average v	17.03	19.00	15.06	14.80	16.11	14.90	16.56	15.93	19.81	22.83
Max v	57	97	23	21	34	42	29	43	118	37
Total ev	596	671	141	120	163	313	88	409	1198	87
Average ev	9.17	10.82	7.83	6.00	9.06	7.45	9.78	7.57	5.76	14.50
Max ev	44	97	16	15	34	29	23	43	59	36
Total iv	845	935	232	217	218	507	136	762	3813	102
Average iv	13.00	15.08	12.89	10.85	12.11	12.07	15.11	14.11	18.33	17.00
Max iv	57	33	23	21	27	42	27	33	105	27

CONCLUSIONS

As articulated earlier, the objective achieved by this project was an analysis of software code produced by the Kuali Foundation for universities and colleges. Since it was conceived in 2004, the Kuali Foundation has successfully implemented six enterprise software modules for higher education on a timely, cost-effective manner including:

- Financial System (2005): Financial software that meets the needs of all Carnegie Class institutions
- Coeus (2006): Research administration for grants administration to federal funding agencies
- Student (2007): Business needs of students, faculty and institutions throughout the academic lifecycle
- Rice (2007): Middleware products that integrate products allowing applications to be built in an agile fashion
- Open Library Environment (2010): Integration of academic and research libraries for managing and delivering intellectual information
- Mobility (2011): Connect mobile devices to a variety of campus systems

Prior to the 2012 study, no research had been completed of the Kuali software analyzing the code quality. In that study four Financial System modules were analyzed. In this study, three Kauli applications (Financial Systems, Rice, and Coeus) containing 27 modules were analyzed for their quality attributes. McCabe IQ was used to determine the McCabe metrics including cyclomatic, essential, and module design complexities. Overall, the code quality for Financial System and Rice was found to be good quality with about 98.5% of the software methods exhibiting cyclomatic complexities less than or equal to 10. Coeus exhibited a higher average v = 2.00 which borders on the McCabe threshold for low application quality.

There are low quality methods. When high risks methods were examined, the average v, ev, and iv climbed measurably. Overall, the findings showed that risker software as a percentage of total software methods for the four studied methods is as follows:

- Kauli Financial System - 1.41%
- Kauli Rice - 1.41%
- Kauli Coeus - 2.00%

This research shows there are low quality methods in the Kauli applications, specifically in Kauli Coeus. Two Coeus methods, Budget and Grants.govS2S Submission, have a poor ratio of high risk to total methods. These two modules also have riskier high average v, ev, and iv. Since Grants.govS2S Submission is granularly very large, these negative McCabe metrics are potentially troubling. For high risk modules, high average and individual v and ev were observed. Coeus lead the way with high averages (mean v and mean ev of 22.83 and 14.50, respectively). Coeus also had the highest individual method v and ev of 97 and 97, respectively. Overall, the Kuali Foundation continues to serve the university community with enterprise software targeted specifically for university and college business models, and it is delivering cost-effective, quality processes to the higher education community. There are, however, pockets of low quality that should be assessed, monitored, and managed to avoid long term quality degradation.

REFERENCES

Aberdour, M. (2007). Achieving Quality in Open Source Software. *IEEE Software*, 58-64.

Butler, C. W. (2012). The Role of Community Source Software in University Business Models. *Journal of Higher Education Theory and Practice, 12*(6), 9.

CURL Staff. (05, 14 2010). *MIDESS Functiona and Technical Requirements Specification.* Retrieved 2011, from http://ludos.leeds.ac.uk/midess/MIDESS%20workpackage%202%20-%20Functional%20and%20Technical%20Requirements%20Specification.pdf

Hanganu, G. (2011, 03 11). *OSS Watch Open Source Software Advisory Service.* Retrieved 2011, from www.oss-watch.ac.uk: http://www.oss-watch.ac.uk/resources/communitysource.xml

Jeffery, M. (n.d.). *Northwestern University.* Retrieved from http://www.kellogg.northwestern.edu/: http://www.kellogg.northwestern.edu/faculty/jeffery/htm/publication/ROIforITProjects.pdf

Kaur, K., Minhas, K., Mehan, N., & Kakkar, N. (2009). Static and Dynamic Complexity Analysis of Software Metrics. *World Academy of Science, Engineering and Technology*, 159-161.

Koha. (1999). *Koha*. Retrieved from http://koha.org/: http://koha.org/

Kuali. (2010). *Kuali - OLE*. Retrieved from www.kuali.org: http://www.kuali.org/ole

Kuali Foundation. (2014). *Kuali Foundation*. Retrieved from www.kuali.org: http://www.kuali.org/kc

McCabe Staff. (n.d.). *McCabe Demostration Page.* Retrieved 2011, from www.mccabe.com: http://www.mccabe.com/contact_iqDemo.htm

McCabe Staff. (n.d.). *McCabe IQ Glossary of Terms*. Retrieved from www.mccabe.com: http://www.mccabe.com/iq_research_iqgloss.htm#t

McCabe Staff. (n.d.). *McCabe Metrics.* Retrieved 2011, from www.mccabe.com: http://www.mccabe.com/pdf/McCabe%20IQ%20Metrics.pdf

NASA IV&V Facility. (2008, 6 10). *Metrics Data Program - NASA IV&V Facility - Complexity Metrics*. Retrieved 2011, from www.nasa.gov: http://mdp.ivv.nasa.gov/complexity_metrics.html

NASA. (n.d.). *Overview of Software Reliability*. Retrieved 2011, from Goddard Space Flight Center: http://sw-assurance.gsfc.nasa.gov/disciplines/reliability/index.php

NLM Digital Repository Evaluation and Selection Working Group. (2008, 12 02). *Recommendations on NLM Digital Repository Software.* Retrieved 2011, from http://www.nlm.nih.gov/digitalrepository/DRESWG-Report.pdf

Shelly, G. B., & Harry, R. (2010). *Systems Analysis an Design.* Course Technology.

Stoneburner, W. (2010, 03 31). *SMERFS Software Reliability Overview*. Retrieved 2011, from www.slingcode.com: http://slingcode.com/smerfs/overview.php

Technical Council. (2007, 12). *Kuali Standards Version 1.2.* Retrieved 2011, from http://kuali.org: http://kuali.org/files/pdf/KualiStandards.pdf

Traylor, P. S. (2006, 02 13). *http://www.infoworld.com.* Retrieved 2011, from InfoWorld: http://www.infoworld.com/print/20676

Wheeler, B. (2007). Open Source 2010 Reflections on 2007. *EDUCAUSE Review,* 49-67.

Wheeler, B., & DeStefano, J. (n.d.). *Mitigating the Risks of Big Systems.* Retrieved from http://www.nacubo.org/: http://www.nacubo.org/Business_Officer_Magazine/Magazine_Archives/July-August_2007/Mitigating_the_Risks_of_Big_Systems.html

Wikipedia. (2010, 08 11). *Wikipedia - The Free Encyclopedia.* Retrieved 2011, from www.wikipedia.org: http://en.wikipedia.org/wiki/Community_source