EVALUATING THE SUSTAINABILITY OF COMPUTATIONAL SCIENCE AND ENGINEERING SOFTWARE: EMPIRICAL OBSERVATIONS

James Willenbring
Sandia National Laboratories
North Dakota State University

Gursimran Walia
Augusta University

SEKE 2022

July 7, 2022

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science
Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
SAND2021-7711 C

Introduction

- Software sustainability is key to Computational Science and Engineering (CSE)
  - Projects often begin as a research activity
    - Research output is the primary objective
    - Many, but not all projects, end up having a long, impactful life
  - Cutting-edge algorithms and language features -> inherent complexity
  - Performance on new supercomputers
- Large codes
- Previous software sustainability studies have not looked at sustainability metrics specifically for CSE software
- Goal: Identify, characterize and reduce barriers to CSE software sustainability

ECP EXASCALE COMPUTING PROJECT

Approach

- Analyze open source CSE software projects
  - Open repository
  - Substantial history
  - Important in CSE community
    - Members of the Extreme-Scale Scientific Software Development Kit (xSDK)
    - 6/7 receive funding through the DOE's Exascale Computing Project (ECP)
      - ECP has a strong interest in software sustainability
  - Large and small (not too small)
  - Variety of primary development institutions
    - Labs
    - Universities
- Gather metrics to identify correlation with aspects of sustainability
  - Code-based and non-code-based metrics
  - Trends in metrics over time

ECP EXASCALE COMPUTING PROJECT

Approach

- Seven projects were studied
  - Projects were anonymized

- Project 1: linear and non-linear solvers and preconditioners
- Project 2: collection of solvers and enabling technologies
- Project 3: distributed memory direct LU solver
- Project 4: algebraic multigrid sparse preconditioners and solvers
- Project 5: dense linear, least squares, eigen, and S.V.D. solvers
- Project 6: finite element and adaptive mesh refinement
- Project 7: sparse linear and nonlinear eigenvalue solvers

Data Analysis and Results

- Metric set 1 collected from
  - GitHub, GitLab, git command line
  - SLOCCount

TABLE 1: Language, SLOC, Contributors & Commits

Package	Language	SLOC	Contribs	Commits
Project 1	C 83%	796123	198	82663
Project 2	C++ 82%	4179781	250	95384
Project 3	C 96%	80752	14	645
Project 4	C 81%	441057	37	11587
Project 5	C++ 45%	317082	51	8086
Project 6	C++ 100%	293062	114	14668
Project 7	C 91%	109559	26	9045

ECP EXASCALE COMPUTING PROJECT

Data Analysis and Results

- Metric set 2 collected from Metrix++
  - Yearly snapshots taken

TABLE 2: Max Cyclomatic Compexity

Package	2017	2018	2019	2020	2021
Project 1	1786	1788	2319	540	540
Project 2	648	648	648	547	547
Project 3	202	204	301	301	301
Project 4	649	765	815	877	913
Project 5	261	261	261	261	261
Project 6	114	114	137	134	238
Project 7	86	86	86	86	83

TABLE 3: Average Cyclomatic Compexity

Package	2017	2018	2019	2020	2021
Project 1	4.63	4.69	4.88	4.69	4.58
Project 2	2.59	2.47	2.46	2.39	2.30
Project 3	12.50	11.97	10.62	10.31	10.36
Project 4	7.14	6.65	6.50	6.48	6.33
Project 5	5.45	5.40	5.27	5.30	5.33
Project 6	2.34	2.33	2.14	2.48	2.49
Project 7	3.95	3.97	4.14	4.11	4.10

TABLE 4: Lines of Code (in 1,000's)

Package	2017	2018	2019	2020	2021
Project 1	483	525	584	613	657
Project 2	2682	3082	3076	3279	3414
Project 3	55	58	79	81	86
Project 4	410	359	365	390	405
Project 5	128	131	136	137	138
Project 6	107	122	154	216	284
Project 7	71	79	84	89	95

TABLE 5: Maintenance index computed by Metrix++ using complexity and lines of code data

Package	2017	2018	2019	2020	2021
Project 1	1.40	1.41	1.43	1.42	1.41
Project 2	1.21	1.18	1.18	1.19	1.18
Project 3	2.35	2.29	2.08	2.05	2.08
Project 4	1.81	1.78	1.77	1.79	1.79
Project 5	1.76	1.71	1.70	1.70	1.71
Project 6	1.21	1.21	1.19	1.24	1.24
Project 7	1.31	1.32	1.33	1.33	1.33

ECP EXASCALE COMPUTING PROJECT

Discussion of Results

- SLOC and Contributors
  - A low number of contributors or high ratio of SLOC/contrib may be a sustainability risk

TABLE 6: SLOC and Contributors

Package	SLOC	contributors	SLOC/contrib
Project 1	796123	198	4021
Project 2	4179781	250	16719
Project 3	80752	14	5768
Project 4	441057	37	11920
Project 5	317082	51	6217
Project 6	293062	114	2571
Project 7	109559	26	4214

EXASCALE COMPUTING PROJECT

Discussion of Results

- Average Complexity
  - Complex codes are harder to maintain
    - Project 3 has significantly higher average complexity

TABLE 3: Average Cyclomatic Compexity

Package	2017	2018	2019	2020	2021
Project 1	4.63	4.69	4.88	4.69	4.58
Project 2	2.59	2.47	2.46	2.39	2.30
Project 3	12.50	11.97	10.62	10.31	10.36
Project 4	7.14	6.65	6.50	6.48	6.33
Project 5	5.45	5.40	5.27	5.30	5.33
Project 6	2.34	2.33	2.14	2.48	2.49
Project 7	3.95	3.97	4.14	4.11	4.10

ECP EXASCALE COMPUTING PROJECT

Discussion of Results

- Maintenance Index
  - More useful for longer-term tracking, moves slowly
  - Project 3 has most significant change in maintenance index
    - Coincides with large increase in code base size

TABLE 4: Lines of Code (in 1,000's)

Package	2017	2018	2019	2020	2021
Project 1	483	525	584	613	657
Project 2	2682	3082	3076	3279	3414
Project 3	55	58	79	81	86
Project 4	410	359	365	390	405
Project 5	128	131	136	137	138
Project 6	107	122	154	216	284
Project 7	71	79	84	89	95

TABLE 5: Maintenance index computed by Metrix++ using complexity and lines of code data

Package	2017	2018	2019	2020	2021
Project 1	1.40	1.41	1.43	1.42	1.41
Project 2	1.21	1.18	1.18	1.19	1.18
Project 3	2.35	2.29	2.08	2.05	2.08
Project 4	1.81	1.78	1.77	1.79	1.79
Project 5	1.76	1.71	1.70	1.70	1.71
Project 6	1.21	1.21	1.19	1.24	1.24
Project 7	1.31	1.32	1.33	1.33	1.33

Discussion of Results

- Metrix++ Hotspot Feature
  - Identifies and counts regions of code exhibiting a metric value above a user-specified threshold
  - Command to identify regions of code in Project 4 with cyclomatic complexity equal to or greater than 500:

```
metrix++ limit --db-file=proj4.2019.lines.complex.maint.db --max-limit=std.code.complexity:cyclomatic:500
```


ECP EXASCALE COMPUTING PROJECT

Discussion of Results

- Advanced Metrix++ Hotspot Features
  - May be useful for CI or code reviews
  - touched
    - Only consider regions of code added or modified between two snapshots
  - trend
    - Only consider touched regions of code for which the metric value in question has gotten worse

  - Command to identify touched regions of code in Project 4 with cyclomatic complexity equal to or greater than 500 (for trend, replace "touched" with "trend"):

    metrix++ limit --db-file=proj4.2019.lines.complex.maint.db --db- file-prev=../2018-05-24/proj4.2018.lines.complex.maint.db --max-limit=std.code.complexity:cyclomatic:500 --warn-mode=touched

ECP EXASCALE COMPUTING PROJECT

Conclusion and Relevance to Industry

- Next steps
  - Explore sustainability factors not reflected in source code
    - Sustainability of software dependencies
    - Sustainability of the CSE software ecosystem
  - Expand Metrix++ hotspot analysis
    - Use for code reviews
  - Continue study of contributor metrics
  - Tools for Data Mining and Code Analysis Software Development Kit (SDK)

ECP EXASCALE COMPUTING PROJECT

Conclusion and Relevance to Industry

- Metrics provide quantitative data that can be used to support decisions
  - Single metrics should not be used to make broad conclusions
    - For example, code A is more sustainable than code B
  - Changes in metrics can inform if a project is becoming more or less sustainable
    - Evaluate changes in development practices or tools
- An important goal is to identify how to design for sustainability

ECP EXASCALE COMPUTING PROJECT