02 INFORMATION ABOUT PRINCIPAL INVESTIGATORS/PROJECT DIRECTORS(PI/PD) and co-PRINCIPAL INVESTIGATORS/co-PROJECT DIRECTORS

Submit only ONE copy of this form **for each PI/PD and co-PI/PD** identified on the proposal. The form(s) should be attached to the original proposal as specified in GPG Section II.B. Submission of this information is voluntary and is not a precondition of award. This information will not be disclosed to external peer reviewers. ***DO NOT INCLUDE THIS FORM WITH ANY OF THE OTHER COPIES OF YOUR PROPOSAL AS THIS MAY COMPROMISE THE CONFIDENTIALITY OF THE INFORMATION.***

PI/PD Name: Philip Johnson

Gender: ☒ Male ☐ Female

Ethnicity: (Choose one response) ☐ Hispanic or Latino ☒ Not Hispanic or Latino

Race: (Select one or more)
☐ American Indian or Alaska Native
☐ Asian
☐ Black or African American
☐ Native Hawaiian or Other Pacific Islander
☐ White

Disability Status: (Select one or more)
☐ Hearing Impairment
☐ Visual Impairment
☐ Mobility/Orthopedic Impairment
☐ Other
☒ None

Citizenship: (Choose one) ☒ U.S. Citizen ☐ Permanent Resident ☐ Other non-U.S. Citizen

Check here if you do not wish to provide any or all of the above information (excluding PI/PD name): ☒

REQUIRED: Check here if you are currently serving (or have previously served) as a PI, co-PI or PD on any federally funded project ☒

Ethnicity Definition:
Hispanic or Latino. A person of Mexican, Puerto Rican, Cuban, South or Central American, or other Spanish culture or origin, regardless of race.
Race Definitions:
American Indian or Alaska Native. A person having origins in any of the original peoples of North and South America (including Central America), and who maintains tribal affiliation or community attachment.
Asian. A person having origins in any of the original peoples of the Far East, Southeast Asia, or the Indian subcontinent including, for example, Cambodia, China, India, Japan, Korea, Malaysia, Pakistan, the Philippine Islands, Thailand, and Vietnam.
Black or African American. A person having origins in any of the black racial groups of Africa.
Native Hawaiian or Other Pacific Islander. A person having origins in any of the original peoples of Hawaii, Guam, Samoa, or other Pacific Islands.
White. A person having origins in any of the original peoples of Europe, the Middle East, or North Africa.

WHY THIS INFORMATION IS BEING REQUESTED:

The Federal Government has a continuing commitment to monitor the operation of its review and award processes to identify and address any inequities based on gender, race, ethnicity, or disability of its proposed PIs/PDs. To gather information needed for this important task, the proposer should submit a single copy of this form for each identified PI/PD with each proposal. Submission of the requested information is voluntary and will not affect the organization's eligibility for an award. However, information not submitted will seriously undermine the statistical validity, and therefore the usefulness, of information recieved from others. Any individual not wishing to submit some or all the information should check the box provided for this purpose. (The exceptions are the PI/PD name and the information about prior Federal support, the last question above.)

Collection of this information is authorized by the NSF Act of 1950, as amended, 42 U.S.C. 1861, et seq. Demographic data allows NSF to gauge whether our programs and other opportunities in science and technology are fairly reaching and benefiting everyone regardless of demographic category; to ensure that those in under-represented groups have the same knowledge of and access to programs and other research and educational oppurtunities; and to assess involvement of international investigators in work supported by NSF. The information may be disclosed to government contractors, experts, volunteers and researchers to complete assigned work; and to other government agencies in order to coordinate and assess programs. The information may be added to the Reviewer file and used to select potential candidates to serve as peer reviewers or advisory committee members. See Systems of Records, NSF-50, "Principal Investigator/Proposal File and Associated Records", 63 Federal Register 267 (January 5, 1998), and NSF-51, "Reviewer/Proposal File and Associated Records", 63 Federal Register 268 (January 5, 1998).

List of Suggested Reviewers or Reviewers Not To Include (optional)

SUGGESTED REVIEWERS:
Not Listed

REVIEWERS NOT TO INCLUDE:
Not Listed

COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 04-23

NSF 05-576 **06/20/05**

FOR NSF USE ONLY

NSF PROPOSAL NUMBER

FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)

CCF - COMPUTING PROCESSES & ARTIFACT

DATE RECEIVED	NUMBER OF COPIES	DIVISION ASSIGNED	FUND CODE	DUNS# (Data Universal Numbering System)	FILE LOCATION
				965088057	

EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)

996000354

SHOW PREVIOUS AWARD NO. IF THIS IS
☐ A RENEWAL
☐ AN ACCOMPLISHMENT-BASED RENEWAL

IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES ☐ NO ☒ IF YES, LIST ACRONYM(S)

NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE

University of Hawaii

AWARDEE ORGANIZATION CODE (IF KNOWN)

0016105000

ADDRESS OF AWARDEE ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE

University of Hawaii
2530 Dole Street
Honolulu, HI. 968222225

NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE

PERFORMING ORGANIZATION CODE (IF KNOWN)

ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE

IS AWARDEE ORGANIZATION (Check All That Apply) (See GPG II.C For Definitions)
☐ SMALL BUSINESS ☐ MINORITY BUSINESS
☐ FOR-PROFIT ORGANIZATION ☐ WOMAN-OWNED BUSINESS

☐ IF THIS IS A PRELIMINARY PROPOSAL THEN CHECK HERE

TITLE OF PROPOSED PROJECT **A continuous, evidence-based approach to discovery and assessment of software engineering best practices**

REQUESTED AMOUNT	PROPOSED DURATION (1-60 MONTHS)	REQUESTED STARTING DATE	SHOW RELATED PRELIMINARY PROPOSAL NO. IF APPLICABLE
$ **491,096**	**36** months	**09/01/05**	

CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW

☐ BEGINNING INVESTIGATOR (GPG I.A)
☐ DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.C)
☐ PROPRIETARY & PRIVILEGED INFORMATION (GPG I.B, II.C.1.d)
☐ HISTORIC PLACES (GPG II.C.2.j)
☐ SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.1)
☐ VERTEBRATE ANIMALS (GPG II.D.5) IACUC App. Date _____
☐ HUMAN SUBJECTS (GPG II.D.6)
Exemption Subsection _____ or IRB App. Date _____
☐ INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES INVOLVED (GPG II.C.2.j)
_____
☐ HIGH RESOLUTION GRAPHICS/OTHER GRAPHICS WHERE EXACT COLOR REPRESENTATION IS REQUIRED FOR PROPER INTERPRETATION (GPG I.E.1)

PI/PD DEPARTMENT
Dept of Information & Computer Science

PI/PD POSTAL ADDRESS

Honolulu, HI 96822
United States

PI/PD FAX NUMBER
808-956-3548

NAMES (TYPED)	High Degree	Yr of Degree	Telephone Number	Electronic Mail Address
PI/PD NAME **Philip Johnson**	**Ph.D.**	**1990**	**808-956-3489**	**johnson@hawaii.edu**
CO-PI/PD				
CO-PI/PD				
CO-PI/PD				
CO-PI/PD				

CERTIFICATION PAGE

Certification for Authorized Organizational Representative or Individual Applicant:

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 04-23. Willful provision of false information in this application and its supporting documents or in reports required under an ensuing award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflicts which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

Drug Free Work Place Certification

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Drug Free Work Place Certification contained in Appendix C of the Grant Proposal Guide.

Debarment and Suspension Certification

(If answer "yes", please provide explanation.)

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency? Yes ☐ No ☒

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Debarment and Suspension Certification contained in Appendix D of the Grant Proposal Guide.

Certification Regarding Lobbying

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding $100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding $150,000.

Certification for Contracts, Grants, Loans and Cooperative Agreements

The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, ''Disclosure of Lobbying Activities,'' in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all tiers including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, Title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than $10,000 and not more than $100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE	SIGNATURE	DATE
NAME		

TELEPHONE NUMBER	ELECTRONIC MAIL ADDRESS	FAX NUMBER

*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

Project Summary

This research proposal presents a new, evidence-based approach to the generation and adoption of best practices. Instead of looking outward into the community for best practices, and attempting to adapt them to one's own environment, our research will investigate how best practices can emerge organically from within one's current organizational and project context. Instead of relying on politics or persuasiveness for adoption, our research approach involves instrumentation that generates empirical data that can be used to either argue for the benefits of adoption, or else provide evidence that the practice is not actually effective in the current context. Finally, our research will involve analytic approaches designed to generate candidate best practices from analysis of process and product data. To accomplish this, we will synthesize and extend three streams of research: (a) software project telemetry, which provides a mechanism for in-process monitoring of software engineering data streams; (b) software development stream analysis, a mechanism for recognition of micro-processes (such as test-driven design) from low-level developer behaviors), and (c) episode discovery, a collection of data mining techniques for discovery of patterns in event data. Our project has the following specific objectives:

(1) Enhancement of the Software Development Stream Analysis mechanism to support a variety of current best practices, and determination of the kinds of best practices that are amenable (or not amenable) to recognition using SDSA.

(2) Development of integration mechanisms between SDSA and Software Project Telemetry in order to allow users to determine how practices recognized by SDSA relate to telemetry data any particular point in time.

(3) Development of an episode discovery subsystem in Hackystat to support automated recognition of patterns in developer behavior.

(4) Classroom-based, case study evaluation of the proposed techniques. In addition to providing initial data regarding the effectiveness of our approach, classroom use will help us to refine the technology, develop curriculum materials, and ready the approach for industrial evaluation.

(5) Industry-based evaluation of the proposed techniques. Following classroom evaluation, we will embark on at least one industry case study with the goal of assessing the suitability of this technique in a "real-world" environment.

(6) Packaging of the system and methods for widespread dissemination. We will continue the process we have followed with the Hackystat Project of making our technology and results continuously available to the software engineering community through frequent stable releases of the system and read-only access to our CVS repositories for the latest changes.

(7) Development of curriculum materials regarding continuous, evidence-based discovery and assessment of software engineering best practices. As with Hackystat, we will develop software engineering curriculum materials and assignments that enable the study and analysis of this approach in academic settings.

The intellectual merit of this research includes the application of novel data gathering and analysis techniques for the discovery and evaluation of software engineering best practices, and the evaluation of this technique through classroom and industrial case studies.

The broader impact of this research includes the development of a sophisticated, freely available, open source software system for use by researchers and practitioners to study software engineering best practices, along with associated curriculum materials to support education and technology transfer. As the University of Hawaii is a university with 75% minority students in an EPSCOR state, this project will provide novel research opportunities to underrepresented groups.

TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated. Complete both columns only if the proposal is numbered consecutively.

1 Overview

1.1 Motivation

As with baseball, physics, music, and other skillful human endeavors, there is a vast range of ability associated with software development. For almost 40 years, software development researchers have been attempting to understand, measure, and support the development of superior skill in software development. Sackman performed the seminal research on programmer productivity in 1967, in which he reported a 28:1 difference between the slowest and fastest programmers on a programming task [34]. Subsequent research by Prechelt on Sackman's original dataset in combination with other published datasets indicates a smaller but still significant multiple—from 2:1 to 6:1 depending upon conditions and the kind of statistical comparison used [31].

While comparison of different programmer's effort on a common task is the most direct way to detect productivity variability, it is not the only way. One alternative employs the COCOMO II cost estimation model [4]. COCOMO uses a dataset of approximately 160 completed industrial projects to calibrate a model that computes the effort required to complete a project based upon characteristics of the software to be developed and the organization doing the development. In COCOMO, the effort differential between best and worst programming teams with respect to capability is 3.53, applications experience is 1.51, language and tools experience is 1.43, platform experience is 1.40, and team cohesion is 1.29. Multiply these together, and the COCOMO model indicates a theoretical productivity difference of 13:1 between the most suited and least suited programming teams for a given software project.

Of course, one can also argue that there is infinite variability between programmers, since certain kinds of programming tasks are so challenging that some programmers will never complete them no matter how much effort they invest. For example, in a private correspondence with a former member of a tool development team for a major vendor, he stated that he viewed his product as inferior to a competitor's and that it would most likely would remain so regardless of the level of resources expended by his company, basically because the competitor's product development was led by a world class designer. In summary, while the actual multiple can be debated, the presence of substantial programmer variability cannot.

Programmer variability creates two basic kinds of challenges for the software engineering research community: (1) How can we raise the average productivity of software developers, and (2) How can we reduce the variability between the best and worst software developers? In general, we have responded to these challenges in one or more of three ways: through abstraction, automation, or best practices.

The evolution of programming languages from machine language to assembly language to high level languages to executable specification languages exemplifies the successful application of abstraction to improving the average productivity of software developers by reducing the amount and complexity of code required to accomplish a given task. A single keyword such as "synchronized" in a high level language like Java might require thousands of lines of code to implement correctly in assembly language. Indeed, software disasters such as the Therac-25 were ultimately attributed to incorrect implementation of synchronization in custom software in a low-level language [28].

Automation refers to the development of scripts or other approaches to ensuring that a sequence of development tasks are carried out consistently, reliably, and correctly. One example is an automated daily build mechanism, which might (a) create a "clean" initial build state, (b) check out the latest version of a system from a configuration management repository, (c) compile the latest version, (d) deploy the latest version to a run-time environment (such as installation on a web server), (e) run all functional (i.e. unit) and non-functional (i.e. load) tests associated with the latest version, (g) build the documentation associated with the latest version, (h) generate a report associated with the build process, and (i) email results to developers and managers.

The difference between abstraction and automation is that abstraction creates a "black box" while automation does not. For example, the implementation of the synchronized keyword in Java is a black box: no application developer would be expected to maintain or debug this language construct and, in general, developers can simply assume that this abstraction functions correctly. A daily build script which is developed, debugged, and maintained by developers does not provide abstraction but nevertheless provides important benefits as a form of automation: it can vastly reduce the productivity impact of developers not carrying out the sequence of actions required to build the product correctly, or even the productivity impact of not building the system at all due to the time, overhead, and tedium associated with the activities.

While abstraction is the province of languages and other expressive media, and automation is the province of tools and environments, best practices focus on the behavior and activities of people during software development. The seminal software engineering best practice is the waterfall lifecycle model, which was first described in the early 1970's and provided an efficient and effective partitioning of development into a sequence of phases: specification, design, implementation, testing, and maintenance. Provided that system requirements can be specified in advance and are guaranteed not to change, the waterfall lifecycle model still constitutes a viable best practice for software engineering.

The Software Engineering Body of Knowledge (SWEBOK) illustrates the variety of forms of best practice [1]. SWEBOK provides a map to the state of the art in software engineering, and divides the landscape into ten areas: requirements, design, construction, testing, maintenance, engineering management, configuration management, process, tools, and quality. The SWEBOK material shows that abstraction, automation, and best practices are not independent concepts but are instead deeply entwined: best practices (such as testing) engender new forms of abstraction (formal languages for testing) and automation (tools for automated test definition and/or invocation). Conversely, new tools (such as automated test frameworks) can catalyze new best practices (such as test driven design).

One might naively assume that becoming a world class software development group would require nothing more than downloading the SWEBOK and implementing all of its best practices and the abstractions and automations that they require. Unfortunately, software engineering best practices are highly contextual: a practice that provides immense benefits in one organizational culture and development context could prove disastrous in another. For example, a best practice such as Cleanroom might be essential in the development of a complex, life-critical application but too costly to justify in a startup environment where time to market is critical. In addition, software engineering best practices can be in conflict. The Extreme Programming [3] best practice eschews the use of the Code Inspection [7] best practice, claiming that the use of Pair Programming obviates the need for a separate inspection activity.

The context sensitivity of software engineering best practices creates a number of problems. First, how can an organization improve by adoption of best practices when it is so difficult to determine their appropriateness? Some organizations might address this problem via a trial-and-error approach, where various best practices are "tried on for size". Others might hire consultants to tell the organization which practices to adopt. Still others might utilize models for process improvement such as the CMMI [33], which could be viewed as "best practices for adopting best practices".

Second, how do "best practices" actually become recognized as such? For example, the best practice of "Extreme Programming" would likely have become a forgotten experiment in an alternative software development process at Chrysler Corporation had Kent Beck not decided to vigorously market the approach with books, lectures, and networking. Ironically, the project on which XP based its initial claims for success was eventually canceled without fulfilling its requirements and is now used as evidence against XP by its detractors [25].

In summary, software engineering uses three methods to address the problem of programmer productivity variability: abstraction, automation, and best practices. Unfortunately, the creation of best practices, and their adoption into new contexts is traditionally mediated by political and social processes that may be

quite unrelated to the actual effectiveness of the practice and its associated abstractions/automations in the organization.

1.2 Approach

This research proposal presents a new, evidence-based approach to the generation and adoption of best practices. Instead of looking outward into the community for best practices, and attempting to adapt them to one's own environment, our research will investigate how best practices can emerge organically from within one's current organizational and project context. Instead of relying on politics or persuasiveness for adoption, our research approach involves instrumentation that generates empirical data that can be used to either argue for the benefits of adoption, or else provide evidence that the practice is not actually effective in the current context. Finally, our research will involve analytic approaches designed to generate candidate best practices from data mining techniques applied to process and product data.

This research approach is made possible by our research and development activities over the past four years in Project Hackystat [15], an open source framework for automated collection and analysis of software engineering process and product data. The Hackystat system provides a unique technological and methodological foundation that we will leverage to pursue this research. First, Hackystat implements an automated approach to metrics collection by attaching sensors to development tools. This makes it possible to capture both low and high-level data about processes and products with a level of precision and completeness not possible with manual approaches. Second, Hackystat provides an implementation of Software Project Telemetry, an approach to in-process monitoring, analysis, and decision-making based upon the generation of high-level abstractions of the sensor data stream. Software Project Telemetry provides a means to understand whether the overall project trajectory is stable, improving, or declining at a particular point in time. Third, Hackystat provides a prototype implementation of Software Development Stream Analysis, which observes the low-level behaviors (i.e. practices) of individuals and classifies them in various ways. For example, SDSA can be used to identify when a developer is using test-driven design.

The combination of Hackystat, Software Project Telemetry, and Software Development Stream Analysis provide a mechanism for emergent, context-sensitive evaluation of best practices within an organization. For example, an organization using Hackystat on a project can use Software Project Telemetry to establish baseline values for various software development measures. Software Development Stream Analysis provides a way to characterize the practices of developers as they perform development activities. Integrating these two analyses together provides a way to relate the practices of developers to their outcomes in terms of process and product measures. So, for example, if a developer decides to switch to the use of pair programming on a trial basis, they can see if this new practice makes an impact on the measures of process and product captured by Software Project Telemetry. Conversely, if Software Project Telemetry reveals a significant decline in process or product metrics (such as a drop in the test case coverage of the system), the Software Development Stream Analyses can be used to assess whether some change in practice could be responsible (such as a change from test-first to test-last design).

Hackystat, Software Project Telemetry, and Software Development Stream Analysis together form an empirical, low-cost, and in-process approach to assessing best practices when the practices are known a priori and recognition rules for them can be built into the SDSA system. The final component of this research will investigate approaches to the discovery of new best practices. To do this, we will incorporate recent research in data mining known as "episode discovery" [11, 29, 2]. The goal of episode discovery is to uncover behavioral patterns in a stream of time-stamped data. For example, in a "smart house", if an occupant repeatedly turns on a light after opening the front door, the episode discovery mechanism should discover this pattern after a number of repetitions, thus enabling the smart house to begin automatically turning on the light whenever the front door is opened by this occupant. In this research, we will use episode

discovery to uncover repeated patterns in developer behavior, and correlate them to the software project telemetry data. As a simple example, episode discovery might find that one developer consistently writes and executes tests against their code prior to committing it to the CVS repository, and that this pattern is correlated with significantly less daily build failures attributed to this developer.

Our approach compares in interesting ways to the more traditional approach to evaluation of best practices. Both cases involve a trial adoption of the best practice, the collection of data on the effect of the practice, and an eventual assessment of efficacy of the practice. However, the traditional approach typically involves a "one off" experiment on a sample project with specialized data collection during the project, and analysis of the success or failure of the practice once the project is concluded. In contrast, our approach involves the introduction of sensor-based instrumentation into the development environment, which allows in-process collection and analysis of data concerning the practice. This has two significant implications. First, the presence of automated metrics collection and analysis allows for more "opportunistic" evaluation of new practices at any point in a project's lifecycle: there is no need to create a special project with special instrumentation for evaluation purposes. Second, the presence of instrumentation enables a "bottom-up" approach to best practice discovery, in which the behaviors of successful practitioners can be analyzed and compared to others to determine if there are generalizable routines, or patterns, that can be articulated.

1.3 Objectives

The overall objective of this research is to design, implement, and evaluate a continuous, evidence-based approach to in-process discovery and assessment of context-sensitive best practices during software development. This overall objective has the following sub-objectives:

- Enhancement of the Software Development Stream Analysis mechanism to support a variety of current best practices, and determination of the kinds of best practices that are amenable (or not amenable) to recognition using SDSA.
- Development of integration mechanisms between SDSA and Software Project Telemetry in order to allow users to determine how practices recognized by SDSA relate to telemetry data any particular point in time.
- Development of an episode discovery subsystem in Hackystat to support automated recognition of patterns in developer behavior.
- Classroom-based, case study evaluation of the proposed techniques. In addition to providing initial data regarding the effectiveness of our approach, classroom use will help us to refine the technology, develop curriculum materials, and ready the approach for industrial evaluation.
- Industry-based evaluation of the proposed techniques. Following classroom evaluation, we will embark on at least one industry case study with the goal of assessing the suitability of this technique in a "real-world" environment.
- Packaging of the system and methods for widespread dissemination. We will continue the process we have followed with the Hackystat Project of making our technology and results continuously available to the software engineering community through frequent stable releases of the system and read-only access to our CVS repositories for the latest changes.
- Development of curriculum materials regarding continuous, evidence-based discovery and assessment of software engineering best practices. As with Hackystat, we will develop software engineering curriculum materials and assignments that enable the study and analysis of this approach in academic settings.

2 Related Work

2.1 Hackystat

For the past several years, we have been developing a framework for automated software development process and product metric collection and analysis called Hackystat. This framework differs from other approaches to software product and process measurement in one or more of the following ways:

- Hackystat uses sensors to unobtrusively collect data from development environment tools; there is no chronic overhead on developers to collect product and process data. In contrast, tools such as the Process Dashboard [32] involve manual data collection.
- Hackystat is tool, environment, process, and application agnostic. The architecture does not suppose a specific operating system platform, a specific integrated development environment, a specific software process, or specific application area. A Hackystat system is configured from a set of modules that determine what tools are supported, what data is collected, and what analyses are run on this data. In contrast, tools such as TSP Tool [6] implement support for a fixed set of metrics under a fixed process on a single platform
- Hackystat is intended to provide in-process project management support. Traditional software metrics approaches, such as the NASA Metrics Data Program [5], are based upon the "project repository" method, in which data from prior completed projects are used to make predictions about or support control of a current project. In contrast, Hackystat is designed to collect data from a current, ongoing project, and use that data as feedback into the current project.
- Hackystat is open source and is available to the academic and commercial software development community for no charge. In contrast, commercial toolkits are closed source and have licensing fees.

The design of Hackystat [18] has resulted from of prior research in our lab on software measurement, beginning with research into data quality problems with the PSP [17] and which continued with the LEAP system for lightweight, empirical, anti-measurement dysfunction, and portable software measurement [21].

To use Hackystat, the project development environment is instrumented by installing Hackystat sensors, which developers attach to the various tools such as their editor, build system, configuration management system, and so forth. Once installed, the Hackystat sensors unobtrusively monitor development activities and send process and product data to a centralized web service. If a user is working offline, sensor data is written to a local log file to be sent when connectivity can be established with the centralized web service. Project members can then log in to the web server to see the collected raw data and run analyses that integrate and abstract the raw sensor data streams into telemetry. Hackystat also allows project members to configure "alerts" that watch for specific conditions in the sensor data stream and send email when these conditions occur. Figure 1 illustrates the basic architecture of the system.

Hackystat is an open source project. Its sources, binaries, and documentation are available at http://www.hackystat.org. We also maintain a public server running the latest release of the system at http://hackystat.ics.hawaii.edu. Hackystat has been under active development for approximately four years, and currently consists of approximately 1500 classes and 95,000 lines of code. Sensors are available for a variety of tools including Eclipse, Emacs, JBuilder, Jupiter, Jira, Visual Studio, Ant, JUnit, JBlanket, CCCC, DependencyFinder, Harvest, LOCC, Office, and CVS.

Hackystat is being used in a variety of academic and industrial contexts. At the University of Hawaii, Hackystat has been tightly integrated into the undergraduate and graduate software engineering curriculum,

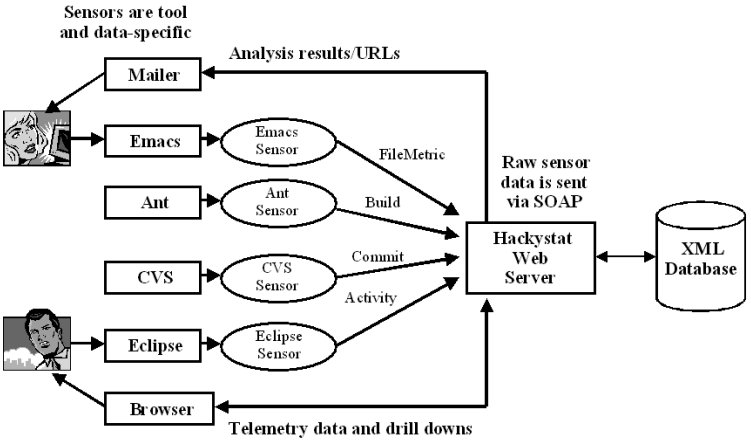

Figure 1. The basic architecture of Hackystat. Sensors are attached to tools directly invoked by developers (such as Eclipse or Emacs) as well as to tools implicitly manipulated by developers (such as CVS or an automated build process using Ant).

and is regularly used by approximately 100 students per year to support project development [19]. A researcher from the Free University of Bozen came to Hawaii to study the Hackystat system to support their work on PROM [35]. Researchers at the University of Maryland are using Hackystat to support assessment of programmer effort [12]. Hackystat has been installed at NASA's Jet Propulsion Lab and used to analyze the daily build process for the Mission Data System [16]. Finally, Hackystat is being used at SUN Microsystems to support research on high performance computing system development productivity [8].

2.2 Software Project Telemetry

The automated, unobtrusive, and low-cost measurement infrastructure provided by Hackystat enables a new approach to software measurement analysis called "Software Project Telemetry". We define Software Project Telemetry as a style of software engineering process and product collection and analysis which satisfies the following properties:

Software project telemetry data is collected automatically by tools that unobtrusively monitor some form of state in the project development environment. In other words, the software developers are working in a "remote or inaccessible location" from the perspective of metrics collection activities. This contrasts with software metrics data that requires human intervention or developer effort to collect, such as PSP/TSP metrics [13].

Software project telemetry data consists of a stream of time-stamped events, where the time-stamp is significant for analysis. Software project telemetry data is thus focused on evolutionary processes in development. This contrasts, for example, with COCOMO [4], where the time at which the calibration data was collected about the project is not significant.

Software project telemetry data is continuously and immediately available to both developers and managers. Telemetry data is not hidden away in some obscure database guarded by the software quality improvement group. It is easily visible to all members of the project for interpretation.

Software project telemetry exhibits graceful degradation. While complete telemetry data provides the best support for project management, the analyses should not be brittle: they should still provide value even if sensor data occasionally "drops out" during the project. Telemetry collection and analysis should provide decision-making value even if these activities start midway through a project.

Software project telemetry is used for in-process monitoring, control, and short-term prediction. Telemetry analyses provide representations of current project state and how it is changing at the time scales of days, weeks, or months. The simultaneous display of multiple project state values and how they change over the same time periods allow opportunistic analyses—the emergent knowledge that one state variable appears to co-vary with another in the context of the current project.

Software Project Telemetry enables a more incremental, distributed, visible, and experiential approach to project decision-making. For example, if one finds that complexity telemetry values are increasing, *and* that defect density telemetry values are also increasing, then one could try corrective action (such as simplification of overly complex modules) and see if that results in a decrease in defect density telemetry values. One can also monitor other telemetry data to see if such simplification has unintended side-effects (such as performance degradation). Project management using telemetry thus involves cycles of hypothesis generation (Does module complexity correlate with defect density?), hypothesis testing (If I reduce module complexity, then will defect density decrease?), and impact analysis (Do the process changes required to reduce module complexity produce unintended side-effects?). Finally, Software Project Telemetry supports decentralized project management: since telemetry data is visible to all members of the project, it enables all members of the project–developers and managers–to engage in these management activities.

Figure 2 shows an example telemetry report. This report illustrates the relationship between aggregate code churn (the lines added and deleted from the CVS repository by all members of the project) and aggregate build results (the number of build attempts and failures on a given day via invocation of the Ant build tool). Note that Telemetry Reports are always defined without reference to a specific project or time interval. The specification of the project and time interval for presentation in the report is specified when the report is generated, not as part of its definition. Thus, project members can now run this telemetry report over differing sets of days, or else change the time scale to Weeks for Months to see if different trends emerge from these alternative perspectives. In addition, once defined, members of other projects on this server could use the BuildAndChurn telemetry report to see if it adds decision-making value to their project management activities.

2.3 Software Development Stream Analysis

Zorro is built on top of Hackystat platform. Development activity data is grouped together for development streaming, stream tokenization and episode classification. Development stream is divided into small episodes with the help of tokenizer. Each episode is a series of continuous activities that are isolated by token activities. For instance, test-pass episodes are created when there is a successful unit test invocation. All development activities happened between two continuous successful test invocations belong to this test-pass episode. We evaluate episode with pre-defined rules using JESS [10] rule engine system.

Hackystat sensors collect both process of software development and state data of projects. To eclipse IDE sensor collects most development activities such as new project, open project, new file, file open, file close, file edit, refactoring, unit test etc. Same kind of activities are grouped together to make sub development streams. They merge together to make development stream. Activities irrelevant to the project are filtered out in merging process.

2.4 Episode Discovery

Episode discovery related work goes here.

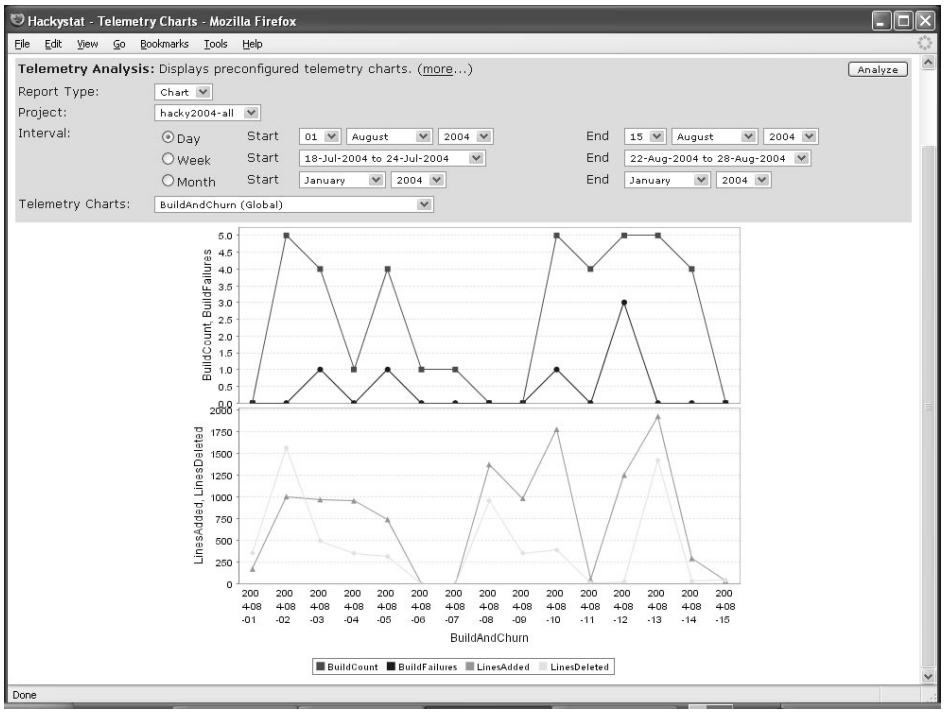

Figure 2. A telemetry report that compares code churn (lines added and lines deleted) to build results (number of build attempts and number of failures.

2.5 Evidence-based software engineering

A recent revolution in medical research involves the introduction of an "evidence-based" paradigm. This paradigm arose in response to two observations: the failure to organize medical research into systematic reviews could cost lives, and the clinical judgment of experts compared unfavorably with the results of systematic reviews. The evidence-based approach is starting to be applied outside of medicine, in fields such as psychiatry, nursing, social policy, education, and software engineering.

Kitchenham has been leading the movement for evidence-based software engineering, organizing workshops on this topic and publishing papers explaining the issues involved in applying evidence-based research techniques to software engineering [27, 26]. She and her collaborators propose a five step method for evidence-based software engineering: (1) Convert the need for information [about a software engineering practice] into an answerable question; (2) Track down the best evidence available for answering the question; (3) Critically appraise that evidence for its validity (closeness to the truth), impact (size of the effect), and applicability (usefulness in software development practice); (4) Integrate the critical appraisal with current software engineering knowledge and stakeholder values [to support decision-making]; (5) Evaluate the effectiveness and efficiency in applying Steps 1-4 and seek ways to improve them for next time. While promising, application of systematic reviews and the integration of empirical software engineering data from multiple sources has been found to be challenging [14].

Our proposed research is designed to provide technology, data, and methodology to evidence-based software engineering. The Hackystat framework, along with the Software Project Telemetry, Software Development Stream, and Episode Discovery applications built on top of it, provides new and useful infrastructure for the creation of evidence regarding a given software engineering practice. Our case studies in the class-

room setting will create new, replicable data that can be used in evidence-based evaluation of the test-driven design, and our industrial case studies will provide similar evidence for high performance computing productivity. Finally, our experiences applying the tools and analyzing the data will result in our recommendations for effective ways to utilize the technologies and assess the data that results.

2.6 Results from prior NSF research

Award number:	CCF02-34568
Program:	Highly Dependable Computing and Communication Systems Research
Amount:	$638,000
Period of support:	September 2002 to September 2006
Title of Project:	Supporting development of highly dependable software through continuous, automated, in-process, and individualized software measurement validation
Principal Investigator:	Philip M. Johnson
Selected Publications:	[22, 30, 20, 19, 18, 16, 24, 9, 23]

The general objective of this research project is to design, implement, and validate software measures within a development infrastructure that supports the development of highly dependable software systems. Contributions of this research project include: (a) development of a specialized configuration of Hackystat to automatically acquire build and workflow data from the configuration management system for the Mission Data System (MDS) project at Jet Propulsion Laboratory; (b) development of analyses over MDS build and workflow data to support identification of potential bottlenecks and process validation; (c) identification of previous unknown variation within the MDS development process; (d) development of a generalized approach to in-process, continuous measurement validation called Software Project Telemetry, (e) substantial enhancements to the open source Hackystat framework, improving its generality and usability; (f) development of undergraduate and graduate software engineering curriculum involving the use of Hackystat for automated software engineering metrics collection and analysis; (g) support for 3 Ph.D., 6 M.S., and 3 B.S. degree students.

3 Research Plan

4 Conclusions

References

[1] Alain Abran and James Moore, editors. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2005.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, Taipei, Taiwan, 1995.

[3] Kent Beck. *Extreme Programming Explained: Embrace Change.* Addison-Wesley, 2000.

[4] Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford Clark, Ellis Horowitz, Ray Madachy, Donald Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[5] Mike Chapman. NASA MDP repository. http://mdp.ivv.nasa.gov/, 2004.

[6] Noopur Davis. Team Software Process tool. http://www.sei.cmu.edu/tsp, 2004.

[7] Michael E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[8] Stuart Faulk, John Gustafson, Philip M. Johnson, Adam A. Porter, Walter Tichy, and Larry Votta. Toward accurate HPC productivity measurement. In *Proceedings of the First International Workshop on Software Engineering for High Performance Computing System Applications*, Edinburgh, Scotland, May 2004.

[9] Stuart Faulk, Philip M. Johnson, John Gustafson, Adam A. Porter, Walter Tichy, and Larry Votta. Measuring HPC productivity. *International Journal of High Performance Computing Applications*, December 2004.

[10] Ernest Friedman-Hill. *JESS in Action*. Mannig Publications Co., Greenwich, CT, 2003.

[11] E. Heierman, G. Youngblood, and D. Cook. Mining temporal sequences to discover interesting patterns. In *Proceedings of the 2004 International Conference on Knowledge Discovery and Data Mining*, Seattle, Washington, 2004.

[12] Lorin Hochstein, Victor Basili, Marvin Zelkowitz, Jeffrey Hollingsworth, and Jeff Carver. Combining self-reported and automatic data to improve effort measurement. In *Proceedings of the 2005 Conference on Foundations of Software Engineering*, 2005.

[13] Watts S. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley, New York, 1995.

[14] A. Jedlitschka and M. Ciolkowski. Towards evidence in software engineering. In *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, 2004.

[15] Philip M. Johnson. Hackystat system. http://www.hackystat.org/.

[16] Philip M. Johnson. The Hackystat-JPL configuration: Overview and initial results. Technical Report CSDL-03-07, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, October 2003.

[17] Philip M. Johnson and Anne M. Disney. The personal software process: A cautionary case study. *IEEE Software*, 15(6), November 1998.

[18] Philip M. Johnson, Hongbing Kou, Joy M. Agustin, Christopher Chan, Carleton A. Moore, Jitender Miglani, Shenyan Zhen, and William E. Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In *Proceedings of the 2003 International Conference on Software Engineering*, Portland, Oregon, May 2003.

[19] Philip M. Johnson, Hongbing Kou, Joy M. Agustin, Qin Zhang, Aaron Kagawa, and Takuya Yamashita. Practical automated process and product metric collection and analysis in a classroom setting: Lessons learned from hackystat-uh. In *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, Los Angeles, California, August 2004.

[20] Philip M. Johnson, Hongbing Kou, Michael G. Paulding, Qin Zhang, Aaron Kagawa, and Takuya Yamashita. Improving software development management through software project telemetry. *IEEE Software*, August 2005.

[21] Philip M. Johnson, Carleton A. Moore, Joseph A. Dane, and Robert S. Brewer. Empirically guided software effort guesstimation. *IEEE Software*, 17(6), December 2000.

[22] Philip M. Johnson and Michael G. Paulding. Understanding HPCS development through automated process and product measurement with hackystat. In *Second Workshop on Productivity and Performance in High-End Computing (P-PHEC)*, February 2005.

[23] Aaron Kagawa. Hackystat MDS supporting MSL MMR. Technical Report CSDL-04-06, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, June 2004.

[24] Aaron Kagawa and Philip M. Johnson. The Hackystat-JPL configuration: Round 2 results. Technical Report CSDL-03-07, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, May 2004.

[25] Gerold Keefer. Extreme programming considered harmful for reliable software development. Technical report, AVOCA GmbH, 2003.

[26] B. Kitchenham. Systematic reviews. In *Proceedings of the 2004 International Symposium on Software Metrics*, 2004.

[27] Barbara Kitchenham, Tore Dyba, and Magne Jorgensen. Evidence-based software engineering. In *Proceedings of the 2004 International Conference on Software Engineering*, 2004.

[28] Nancy Leveson and Clark Turner. An investigation of the Therac-25 accidents. *IEEE Computer*, July 1993.

[29] H. Mannila, H. Toivonen, and A. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the 1995 International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, 1995.

[30] Michael G. Paulding. Measuring the processes and products of HPCS development: Initial results for the optimal truss purpose-based benchmark. Technical Report CSDL-04-13, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii 96822, September 2004.

[31] Lutz Prechelt. The 28:1 Grant/Sackman legend is misleading, or: How large is interpersonal variation really? Technical Report 1999-18, University of Karlsruhe, 1999.

[32] Ken Raisor and David Tuma. Process dashboard for PSP. http://processdash.sourceforge.net/, 2001.

[33] Walker Royce. CMM vs. CMMI: From conventional to modern software management. *The Rational Edge*, February 2002.

[34] H. Sackman, W. Erikson, and E. Grant. Exploratory experimental studies comparing online and offline programming performance. *Communications of the ACM*, 11(1), 1968.

[35] Alberto Sillitti, Andrea Janes, Giancarlo Succi, and Tullio Vernazza. Collecting, integrating and analyzing software metrics and personal software process data. In *Proceedings of the 29th Euromicro Conference*, 2003.

Philip M. Johnson

Information and Computer Sciences
University of Hawaii
1680 East-West Road
Honolulu, HI 96822

(808) 956-3489
fax: (808) 956-3548
johnson@hawaii.edu
http://www.ics.hawaii.edu/∼johnson/

Professional Preparation

Ph.D. in Computer Science, University of Massachusetts. 1990
M.S. in Computer Science, University of Massachusetts. 1985
B.S. in Biology, University of Michigan. 1980
B.S. in Computer Science, University of Michigan. 1980

Appointments

Professor, Information and Computer Sciences, University of Hawaii. 2001—present
Associate Professor, Information and Computer Sciences, University of Hawaii. 1995—2001
Senior Research Fellow, Distributed Systems Technology Centre, University of Queensland, Australia, 1997.
Assistant Professor, Information and Computer Sciences, University of Hawaii. 1990—1995

Publications: Closely Related

P. M. Johnson and H. Kou and M. Paulding and Q. Zhang and A. Kagawa and T. Yamashita, *Improving Software Development Management through Software Project Telemetry*, IEEE Software, August, 2005 (to appear).

S. Faulk and P. M. Johnson and J. Gustafson and A. Porter and W. Tichy and L. Votta, *Measuring HPC Productivity*, International Journal of High Performance Computing Applications, December, 2004.

P. M. Johnson and H. Kou and J. Augustin and C. Chan and C. Moore and J. Miglani and S. Zhen and W. Doane, *Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined*, Proceedings of the 2003 International Conference on Software Engineering, Portland, Oregon, May, 2003.

P. M. Johnson and C. A. Moore and J. A. Dane and R. S. Brewer, *Empirically Guided Software Effort Guesstimation*. IEEE Software, Vol. 17, No. 6, December 2000.

P. M. Johnson and A. M. Disney, *A Critical Analysis of PSP Data Quality: Results from a Case Study*. Journal of Empirical Software Engineering, Volume 4, December, 1999.

Publications: Other Signi£cant

P. M. Johnson and M. Paulding, *Understanding HPCS development through automated process and product measurement with Hackystat*, Second Workshop on Productivity and Performance in High-End Computing (P-PHEC), February, 2005.

P. M. Johnson and H. Kou and J. Augustin and Q. Zhang and A. Kagawa and T. Yamashita, *Practical automated process and product metric collection and analysis in a classroom setting: Lessons learned from Hackystat-UH*, Proceedings of the 2004 International Symposium on Empirical Software Engineering, Los Angeles, California, August, 2004.

P. M. Johnson, M. L. Moffett, and B. T. Pentland, *Lessons Learned from VCommerce: A virtual environment for interdisciplinary learning about software entrepreneurship*, Communications of the ACM, December, 2003.

P. M. Johnson and A. M. Disney, *The Personal Software Process: A Cautionary Case Study*. In IEEE Software, Volume 15, No. 6, November, 1998.

P. M. Johnson, *Design for Instrumentation: High Quality Measurement of Formal Technical Review*. Software Quality Journal, Volume 5, March, 1996.

Synergistic Activities

Editorial Boards:

Journal of Empirical Software Engineering, 2004-present.
IEEE Transactions on Software Engineering, 2000-2004.

Program Chair:

Workshop on Software Engineering for High Performance Computing, 2004, 2005.
International Software Engineering Research Network Annual Meeting, 2000.

Member, Board of Directors:

Hawaii Strategic Development Corporation, 1999-present.
LavaNet, Inc., 2002-present.

Advisory Board Member:

International Journal of Computer Supported Cooperative Work

Program Committees:

XP/Agile Universe, 2004-present.
PROFES 2004-present
International Software Metrics Symposium, 2003-present.
International Symposium on Empirical Software Engineering, 2002-present.
European Conference on Computer Supported Cooperative Work, 1997,1999.
Software Architectures for Cooperative Systems Workshop, 1994
CSCW Tools and Technologies Workshop, 1992,1993

Collaborators

J. Augustin, A, Kagawa, H.Kou, D. Port, M. Paulding, T. Yamashita, Q. Zhang, University of Hawaii.
S. Faulk, University of Oregon
A. Porter, University of Maryland
L. Votta, J. Gustafson, Sun Microsystems
W. Tichy, University of Karlsruhe
D. Reed, University of North Carolina

Thesis Advisor and Postgraduate-Scholar Sponsor

Qin Zhang, University of Hawaii
Carleton Moore, Orincon, Inc.
Danu Tjahjono, Aloha Networks, Inc.
Dadong Wan, Accenture, Inc.
Robert Brewer, LavaNet, Inc.
Mette Moffett, hotU, Inc.
Monir Hodges, Honolulu Community College.
Total graduate students advised: 27

Graduate and Postdoctoral Advisors

Jack Wileden, University of Massachusetts
Victor Lesser, University of Massachusetts
Wendy Lehnert, University of Massachusetts
David Stemple, University of Massachusetts
Daniel Corkill, Blackboard Technology Group

SUMMARY PROPOSAL BUDGET

YEAR 1

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **University of Hawaii**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Philip M Johnson**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Philip M Johnson - Professor**	0.00	0.00	2.00	$ 21,354	$
2.					
3.					
4.					
5.					
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0	
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	0.00	0.00	2.00	21,354	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0	
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0	
3. (2) GRADUATE STUDENTS				54,594	
4. (0) UNDERGRADUATE STUDENTS				0	
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0	
6. (0) OTHER				0	
TOTAL SALARIES AND WAGES (A + B)				75,948	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				14,396	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				90,344	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.)					
equipment item 1 $ 0					
TOTAL EQUIPMENT				0	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				5,000	
2. FOREIGN				0	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ 0					
2. TRAVEL 0					
3. SUBSISTENCE 0					
4. OTHER 0					
TOTAL NUMBER OF PARTICIPANTS (0) TOTAL PARTICIPANT COSTS				0	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				10,000	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0	
3. CONSULTANT SERVICES				0	
4. COMPUTER SERVICES				0	
5. SUBAWARDS				0	
6. OTHER				0	
TOTAL OTHER DIRECT COSTS				10,000	
H. TOTAL DIRECT COSTS (A THROUGH G)				105,344	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **MTDC (Rate: 36.3000, Base: 105344)**					
TOTAL INDIRECT COSTS (F&A)				38,240	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				143,584	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ 143,584	$

M. COST SHARING PROPOSED LEVEL $ 0 AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Philip M Johnson**

ORG. REP. NAME*

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

 ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

SUMMARY PROPOSAL BUDGET YEAR 2

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **University of Hawaii**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Philip M Johnson**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Philip M Johnson - Professor**	0.00	0.00	2.00	$ 23,276	$
2.					
3.					
4.					
5.					
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0	
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	0.00	0.00	2.00	23,276	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0	
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0	
3. (2) GRADUATE STUDENTS				56,778	
4. (0) UNDERGRADUATE STUDENTS				0	
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0	
6. (0) OTHER				0	
TOTAL SALARIES AND WAGES (A + B)				80,054	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				15,009	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				95,063	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.)					
equipment item 1 $ 0					
TOTAL EQUIPMENT				0	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				5,000	
2. FOREIGN				0	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ 0					
2. TRAVEL 0					
3. SUBSISTENCE 0					
4. OTHER 0					
TOTAL NUMBER OF PARTICIPANTS (0) TOTAL PARTICIPANT COSTS				0	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				10,000	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0	
3. CONSULTANT SERVICES				0	
4. COMPUTER SERVICES				0	
5. SUBAWARDS				0	
6. OTHER				0	
TOTAL OTHER DIRECT COSTS				10,000	
H. TOTAL DIRECT COSTS (A THROUGH G)				110,063	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)					
MTDC (Rate: 36.3000, Base: 110063)					
TOTAL INDIRECT COSTS (F&A)				39,953	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				150,016	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ 150,016	$

M. COST SHARING PROPOSED LEVEL $ 0 AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Philip M Johnson**

ORG. REP. NAME*

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

 *ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

SUMMARY PROPOSAL BUDGET YEAR 3

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **University of Hawaii**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Philip M Johnson**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Philip M Johnson - Professor**	**0.00**	**0.00**	**2.00**	$ **25,836**	$
2.					
3.					
4.					
5.					
6. (**0**) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	**0.00**	**0.00**	**0.00**	**0**	
7. (**1**) TOTAL SENIOR PERSONNEL (1 - 6)	**0.00**	**0.00**	**2.00**	**25,836**	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (**0**) POST DOCTORAL ASSOCIATES	**0.00**	**0.00**	**0.00**	**0**	
2. (**0**) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	**0.00**	**0.00**	**0.00**	**0**	
3. (**2**) GRADUATE STUDENTS				**59,049**	
4. (**0**) UNDERGRADUATE STUDENTS				**0**	
5. (**0**) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				**0**	
6. (**0**) OTHER				**0**	
TOTAL SALARIES AND WAGES (A + B)				**84,885**	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				**15,666**	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				**100,551**	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) **equipment item 1** $ **0**					
TOTAL EQUIPMENT				**0**	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				**5,000**	
2. FOREIGN				**0**	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ **0**					
2. TRAVEL **0**					
3. SUBSISTENCE **0**					
4. OTHER **0**					
TOTAL NUMBER OF PARTICIPANTS (**0**) TOTAL PARTICIPANT COSTS				**0**	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				**10,000**	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				**0**	
3. CONSULTANT SERVICES				**0**	
4. COMPUTER SERVICES				**0**	
5. SUBAWARDS				**0**	
6. OTHER				**0**	
TOTAL OTHER DIRECT COSTS				**10,000**	
H. TOTAL DIRECT COSTS (A THROUGH G)				**115,551**	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **MTDC (Rate: 36.3000, Base: 115552)**					
TOTAL INDIRECT COSTS (F&A)				**41,945**	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				**157,496**	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				**0**	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ **157,496**	$

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Philip M Johnson**

ORG. REP. NAME*

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

 ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

SUMMARY PROPOSAL BUDGET

Cumulative

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **University of Hawaii**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Philip M Johnson**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Philip M Johnson - Professor**	**0.00**	**0.00**	**6.00**	$ **70,466**	$
2.					
3.					
4.					
5.					
6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	**0.00**	**0.00**	**0.00**	**0**	
7. (**1**) TOTAL SENIOR PERSONNEL (1 - 6)	**0.00**	**0.00**	**6.00**	**70,466**	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (**0**) POST DOCTORAL ASSOCIATES	**0.00**	**0.00**	**0.00**	**0**	
2. (**0**) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	**0.00**	**0.00**	**0.00**	**0**	
3. (**6**) GRADUATE STUDENTS				**170,421**	
4. (**0**) UNDERGRADUATE STUDENTS				**0**	
5. (**0**) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				**0**	
6. (**0**) OTHER				**0**	
TOTAL SALARIES AND WAGES (A + B)				**240,887**	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				**45,071**	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				**285,958**	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) $ **0**					
TOTAL EQUIPMENT				**0**	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				**15,000**	
2. FOREIGN				**0**	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ **0**					
2. TRAVEL **0**					
3. SUBSISTENCE **0**					
4. OTHER **0**					
TOTAL NUMBER OF PARTICIPANTS (**0**) TOTAL PARTICIPANT COSTS				**0**	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				**30,000**	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				**0**	
3. CONSULTANT SERVICES				**0**	
4. COMPUTER SERVICES				**0**	
5. SUBAWARDS				**0**	
6. OTHER				**0**	
TOTAL OTHER DIRECT COSTS				**30,000**	
H. TOTAL DIRECT COSTS (A THROUGH G)				**330,958**	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)					
TOTAL INDIRECT COSTS (F&A)				**120,138**	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				**451,096**	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				**0**	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ **451,096**	$

M. COST SHARING PROPOSED LEVEL $ **0** AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Philip M Johnson**

ORG. REP. NAME*

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

C ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

Budget Justification

Salaries

The project budget provides salary support for the principal investigator (two summer months) and two graduate research assistants (11 months) for each of the three years. The principal investigator will perform or supervise all major system design enhancements and empirical studies, manage and train the graduate student assistants, and supervise case studies.

Travel

The project budget provides funds for two trips per year from Hawaii to the mainland. These trips will be used to attend relevant conferences (e.g., ICSE, FSE, ISERN, or STAR), where he may report on the results of his own work and obtain first hand information on related work.

Computer Services

The principal investigator directs the Collaborative Software Development Laboratory, which is currently equipped with one enterprise SUN server and 8 workstations, along with a printer and networking peripherals. The project budget provides funds of $10,000/year to provide workstations and associated software for the principal investigator and the two graduate research assistants. In addition, these funds will maintain a server for the public Hackystat site and the developer services site.

Cost breakdown

All calculations are rounded to the nearest dollar.

Summer Salary. Summer salary is calculated as two additional months of the principal investigator's annual nine month salary. Summer salary includes the pay increases as negotiated through collective bargaining. Pay increases are as follows: 5% in 2006 (year 1), 9% in 2007 (year 2) and 11% in 2008 (year 3).

Research Assistants. This cost category provides support for two graduate research assistants for each of the four years. The salary for each research assistant is based upon RA-Step 3.

Fringe benefits. Fringe benefits for salaries are calculated at 3.5% for the principal investigator and 25% for the graduate assistants.

Overhead. University of Hawaii overhead cost is calculated as 36.3% of Modified Total Direct Costs (MTDC), i.e. salaries, travel, and supplies (equipment is excluded).

Current and Pending Support

(See GPG Section II.C.2.h for guidance on information to include on this form.)

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Investigator: **Philip Johnson**	Other agencies (including NSF) to which this proposal has been/will be submitted.

Support: ☒ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Supporting Development of Highly Dependable Software Through Continuous, Automated, In-process, and Individualized Software Measurement Validation**

Source of Support: **NSF**

Total Award Amount: $ **650,000** Total Award Period Covered: **09/01/02 - 08/31/06**

Location of Project: **University of Hawaii**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **2.00**

Support: ☐ Current ☒ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Collaborative Research: Cedar -- Cyberinfrastructure for Empirical Data Analysis and Reuse**

Source of Support: **NSF**

Total Award Amount: $ **629,675** Total Award Period Covered: **09/01/05 - 08/31/09**

Location of Project: **University of Hawaii**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **2.00**

Support: ☐ Current ☒ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **A continuous, evidence-based approach to discovery and assessment of software engineering best practices**

Source of Support: **NSF**

Total Award Amount: $ **451,096** Total Award Period Covered: **09/01/05 - 08/31/08**

Location of Project: **University of Hawaii**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad: **0.00** Sumr: **2.00**

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: $ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: $ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Summ:

*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

FACILITIES, EQUIPMENT & OTHER RESOURCES

FACILITIES: Identify the facilities to be used at each performance site listed and, as appropriate, indicate their capacities, pertinent capabilities, relative proximity, and extent of availability to the project. Use "Other" to describe the facilities at any other performance sites listed and at sites for field studies. USE additional pages as necessary.

Laboratory:

Clinical:

Animal:

Computer: **The Collaborative Software Development Laboratory currently provides a dozen workstation-class computers and three server-class computers for research purposes. The Department of Information and Computer Sciences provides several hundred computers for instructional purposes.**

Office:

Other:

MAJOR EQUIPMENT: List the most important items available for this project and, as appropriate identifying the location and pertinent capabilities of each.

OTHER RESOURCES: Provide any information describing the other resources available for the project. Identify support services such as consultant, secretarial, machine shop, and electronics shop, and the extent to which they will be available for the project. Include an explanation of any consortium/contractual arrangements with other organizations.