Infusing Software Assurance Research Techniques into Use

Thomas Pressburger
NASA Ames Research Center
Moffett Field, CA 94303
650-604-4878
Tom.Pressburger@nasa.gov

Ben Di Vito
NASA Langley Research Center
Hampton, VA 23681
757-864-4833
B.DiVito@nasa.gov

Martin S. Feather
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr
Pasadena, CA 91109
818-354-1194
Martin.S.Feather@jpl.jnasa.gov

Michael Hinchey
NASA Goddard Space Flight Center
Greenbelt, MD 20771
301-286-9057
Michael.G.Hinchey@nasa.gov

Lawrence Markosian
QSS Group, Inc.
NASA Ames Research Center
Moffett Field, CA 94035
650-604-6207
lzmarkosian@email.arc.nasa.gov

Luis C. Trevino
2L Research Corp.
Huntsville, AL
256-509-0196
Trevino@hiwaay.net

Abstract—Research in the software engineering community continues to lead to new development techniques that encompass processes, methods and tools. However, a number of obstacles impede their infusion into software development practices. These are the recurring obstacles common to many forms of research. Practitioners cannot readily identify the emerging techniques that may benefit them, and cannot afford to risk time and effort evaluating and trying one out while there remains uncertainty about whether it will work for them. Researchers cannot readily identify the practitioners whose problems would be amenable to their techniques, and, lacking feedback from practical applications, are hard-pressed to gauge the where and in what ways to evolve their techniques to make them more likely to be successful. This paper describes an ongoing effort conducted by a software engineering research infusion team established by NASA's Software Engineering Initiative to overcome these obstacles.[1,2,3]

TABLE OF CONTENTS

1. INTRODUCTION

Technology infusion – the maturation and transfer of research results into practical use – has long been a desirable yet challenging goal [1]. NASA, like many organizations, can benefit from successful technology infusion. However, technology infusion is often difficult. [2] outlines some of the obstacles to technology infusion within NASA's setting, and proposes some remedies, using microelectronics technologies as examples.

Software engineering is a technology area that is subject to these infusion obstacles. [3] observed this a decade ago (also in a NASA setting). Recognition of the growing prominence of software within the development and operation of NASA spacecraft has led to the establishment of the NASA Software Working Group, the purpose of which is:

> "...to develop and oversee the formulation and implementation of an Agency wide plan to work toward continuous, sustained software engineering process and produce improvements in NASA; and to ensure appropriate visibility of software issues within the Agency" [4].

One of the strategies of this group is to "Improve NASA's software engineering practices through research".

This paper is authored by recent and current members of the team responsible for conducting this strategy, a key element of which is to "Implement and transfer mature software engineering research results and new technologies to operational use within NASA". The infusion team's approach to this is the focus herein.

[1] 0-7803-9546-8/06/$20.00© 2006 IEEE

[2] IEEE Aerospace Conference paper #1506, V6, Dec 28, 2005

[3] Luis Trevino contributed to this work while at NASA Marshall Space Flight Center

Obstacles to software engineering infusion

There are many obstacles to software engineering technology infusion, such as the gap between researchers' and practitioners' concepts of adequate maturity; inadequacy of the NASA Technology Readiness Level (TRL) scale for quantifying the size of this gap; the risk-averse nature of most NASA software developers[4]; and the differing motivation structures for researchers and developers. Rarely are there return-on-investment (ROI) models, competitive analyses or other evidence to show a research product's value in specific development environments. There are many software engineering research products and it's difficult for practitioners to identify, evaluate and track those that may be appropriate for them. The practitioner community is also somewhat fragmented, with many contractors—who develop the majority of NASA-funded software—unaware of NASA-funded software engineering research. Finally, software development for NASA missions takes place in the larger context of project management of the entire mission, wherein there is reluctance to commit scarce resources to try out technologies that haven't been thoroughly proven, and even more reluctance to placing them on any critical path.

The net result of these obstacles is a low rate of infusion of software engineering research results into software development practice. Many research efforts culminate in pilot studies that show promise, but thereafter the technique goes unused, and the researcher switches attention to another avenue of research.

Our approach to overcoming these obstacles

The paper is organized into the following sections explaining the approach that our team follows to try to overcome these obstacles to research infusion.

Section 2, Information Gathering: We identify and assess software engineering research that is of relevance to NASA's software development activities. Included in this is research performed both within and outside of NASA.

Section 3, Information Dissemination: We identify the channels to reach the NASA software practitioners who might benefit from the research techniques. We use these channels to publicize the research techniques among NASA and its contractors' software development teams.

Section 4, Brokering Collaborations: We identify and encourage promising collaborations between researchers and NASA software engineering practitioners. This is helped by the availability of funds specifically devoted to support such collaborations. Our infusion team helps recommend the allocation of this funding to worthy collaborations.

[4] This is particularly true of decision-makers in the human space program. Software development processes at United Space Alliance, for example, which develops Space Shuttle flight and ground software, have been rated at SEI level 5 [5], an indicator of the motivation to produce reliable software. "Most software professionals are resistant to change" [3]

Section 5, Collaborations 2004-2005: We summarize the research collaborations conducted to date.

Section 6: Extracting Lessons Learned: Our team tracks the progress of the funded collaborations, and extracts lessons learned from the aggregation of these experiences. These lessons learned help identify challenges to and success factors for technology transfer in NASA, and help refine our team's approach.

2. INFORMATION GATHERING

Our information gathering efforts aim to identify software engineering research taking place that is relevant to NASA's software development activities. Since our effort was chartered in 2002, we have considered both research performed within NASA, research from outside NASA, and commercial products. Our team consists of members of the software engineering research community from several of the NASA centers and JPL. Their experience and activity within the software engineering milieu give the team a broad awareness of ongoing developments in that arena.

To do this across the entire field of software engineering and the entire range of NASA software development needs would be a large-scale task. However, the team's members spend only part of their time on research infusion; overall, for each of the last three years, our team members' efforts have totaled to approximately 1.5 full-time-equivalents per year. Thus coverage of the entire field of software engineering is significantly beyond our scope. Instead, we have narrowed our focus to software engineering research results that:

(1) Have particular relevance to software assurance.

(2) Can be incorporated into existing software development practices with a minimum of disruption.

(3) Are mid- to high-TRL (Technology Readiness Level) research, demonstrating success on a real project, and ready for use more or less as-is.

(4) Are NASA-funded or related technologies or have been suggested by software developers.

We discuss each of these in more detail:

Software assurance focus

We focus on software engineering techniques that have particular relevance to software *assurance,* namely "the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures. For NASA, this includes the disciplines of software quality (functions of software quality engineering, software quality assurance, and software quality control), software safety, software reliability, software verification and validation, and software independent verification and validation." [6]

This choice of focus is driven by two factors: availability of funding to support collaboration studies in this area, and the nature of NASA's software challenges. We have been able to support collaborations with funding provided by NASA's Software Assurance Research Program [7]. As its name suggests, it has a focus on assurance-related techniques, and is a source of promising research results. NASA's missions impose a particularly stringent need for reliable software, coupled with very limited opportunity to field-test such software in advance, as a result of which everyday software assurance practices are not necessarily sufficient–hence the impetus within NASA to conduct and infuse research in this area. Note that software assurance activities (e.g., code inspections) can, and often are, performed by the developers themselves, so we target the entire software development community, not just software assurance or IV&V personnel.

Evolutionary not revolutionary

We limit our attention to research techniques that can be incorporated with a minimum of disruption into existing software development practices. For example, we include methods that improve the effectiveness of reviews, inspections, code walkthroughs and the like – these are practices generally part of current software development practice at NASA. By way of contrast, we exclude from our consideration research techniques that would require a radical shift in existing practices (e.g., an approach that requires formal specification of the entire software system, or a new programming language that is incompatible with existing platforms and personnel skills). Our narrow focus is motivated by the modest level of effort we are able to bring to bear on research infusion, and should not be construed as a lack of interest by NASA in other software engineering research. Indeed, formal methods, which tends to be revolutionary and requiring a greater cost to introduce, continues to be studied within NASA[5].To the extent that the techniques we encompass detect problems earlier in the software lifecycle, they will not only reduce risk, but may also lead to cost savings. However, techniques whose primary goal is cost or time savings (e.g., product lines) tend to be more revolutionary than evolutionary, and so tend to fall outside our scope.

Our team uses the following criteria to assess prospective techniques. Each technique is ranked qualitatively (High, Medium, Low, or Unknown) against the each of the criteria:

a) What is the range of applicability to NASA projects?
b) Is this an enabler for software that would otherwise be infeasible to develop without this research product?
c) What is the expected improvement in productivity over current techniques?
d) What is the projected cost of installing and applying the research product?
e) What is the risk of failure for technical reasons?
f) How easily can the research product(s) be integrated into a software development project?
g) How much training is required to use the research product?
h) Does the research product depend on widespread utilization within the project/mission/enterprise to fulfill its potential?
i) Does the research product have a good user interface (both for input and output)?
j) Is the research product's development organization (or some other organization) able to provide the required level of support to users of the product?
k) Is the value of the research product clearly apparent to the users during (or shortly after) its application?
l) Is anything about the research product likely to cause resistance among users?

Mid- to High-TRL level

We also limit our attention to just mid- to high-TRL (Technology Readiness Level) [12] research products. We use a definition of TRL specialized to software engineering, and look for techniques that are TRL 6 or higher on this scale. The key maturity requirements are that the research products have been applied to *real—usually NASA—problems*, and are ready for use as-is (or nearly so–for example, we anticipate that the technology providers may well need to assist the practitioners make use of their products in lieu of there being a complete set of user manuals, training materials, etc). Again, this focus is dictated by our modest level of effort (we cannot afford the time to look at everything), coupled with the nature of the funding to support collaborations (which is in modest amounts, sufficient to fund a collaboration study, but not sufficient to support further research). We also consider leading edge COTS tools, for example, those whose development has been funded in part by NASA or other government agencies to address software development issues similar to NASA's.

The combination of these factors that narrow our focus make our task feasible within the level of effort available to us. They also help circumvent some of the concerns that have been expressed (e.g., [13], [14]) on relying solely on TRL measures as a means to assess readiness for technology infusion. For example (from [13]): *"...TRLs leave out such considerations as the degree to which the technology is critical to the overall success of the systems..."*; our assessment's questions such as "b) Is this an enabler..." and "c) What is the expected improvement..." address this issue.

[5] For example, the Robust Software Engineering Group, headed by Michael Lowry at NASA Ames Research Center [8]; the JPL Laboratory for Reliable Software, headed by Gerard Holzmann [9]; the Langley Formal Methods group, headed by Ricky Butler [10], Goddard Space Flight Center's Software Engineering Laboratory, headed by Michael Hinchey [11].

NASA-funded or related technologies, or those that have been suggested by software developers

We were directed by the NASA Software Engineering Initiative to focus on NASA-funded research and related technologies. Examples include several static analyzers, one that is funded by NASA research programs and others that are commercially available. As our customer base has grown, we have increased our efforts to solicit technology suggestions from the NASA community.

3. INFORMATION DISSEMINATION

The next step in software research infusion is to disseminate information about those research techniques to potential beneficiaries – NASA software practitioners. We employ both passive and active means to disseminate information. Passive means are based on web pages that make information available to whoever cares to read it. Active means include following specific pathways that lead to identification of likely practitioners, personal contacts, and annual NASA-wide videoconferences.

Passive dissemination of information

Information on the research techniques that we have identified is posted at the research infusion web site [15]

The research product descriptions are organized into levels of increasing detail: groupings of techniques by life cycle activity (for example, requirements specification and analysis), one-page summaries, three page summaries, and pointers to more extensive material, typically technical papers that the researchers have posted on their own websites. The intent is to help guide the reader to efficiently home in on the techniques that are likely be a good match. Furthermore, the 1- and 3-page summaries uniformly address what the research product is (for example, a tool to detect coding defects without runtime testing), the product's features, its benefits, the successes it's had (where appropriate, focusing on NASA applications), the contexts in which it is best applied, a comparison with alternative products, and a brief discussion of how a successful collaboration should be structured from the perspective of the technology provider.

These are publicly accessible web pages, and so may be located by practitioners within NASA and its contractors by search, or by following links to these pages from various other NASA web pages (for example, the NASA Software Working Group's pages).

Active dissemination of information

Our team members have contacts with NASA software practitioners at their respective centers and with contractors as well. Presumably other NASA software engineering researchers have similar contacts with software practitioners, and might be expected to pursue these to locate likely would-be users of their own techniques, and to serendipitously make connections between practitioners and other research of which they are aware. Our infusion team, through its involvement in gathering information on suitable techniques, has at its fingertips deeper and broader knowledge of those techniques, and so is better able to recognize potential connections. In addition, specific site visits have been conducted to NASA Centers and contractors.

In addition, we have used the NASA Software Working Group (SWG) to spread awareness of its research technologies. The SWG is composed of members from each of the NASA centers, and is in close contact with Software Engineering Process Groups at the centers. This is the kind of channel that few of the NASA software engineering researchers (and even fewer of the non-NASA software engineering researchers) are aware of.

Finally, we hold annual NASA-wide video teleconferences in which we describe the research infusion effort, highlight a crop of promising techniques, and announce a "call for collaboration proposals" (more on this item in the next section). These are aimed at the NASA software practitioner community. Announcements of these are spread through our aforementioned channels, and via various bulletin boards and e-mail lists. Attendance is voluntary, and must therefore compete with the many other demands on software practitioners' time. Thus there is some "self-filtering" by the attendees themselves, to the people who are more likely to be interested/curious/driven to seek improvements, and hence representative of "early adopters" of new ideas. We follow up on their attendance to get their feedback on their level of interest in the showcased technologies, suggestions for new technologies, and software development issues of particular interest. Interested software developers who can't attend the video teleconferences can access online videos; DVDs are also available.

Advantages of our approach

Our efforts serve to increase practitioners' awareness of emerging research techniques. The main advantage our approach has over the status quo derives from our widespread awareness within the research and practitioner communities, and active engagement as brokers between these two communities. In the normal course of events software practitioners have little time to spare to peruse the software research literature, attend research conferences, etc. Similarly, the software researchers themselves are focused primarily on performing their research and keeping abreast of developments at the cutting-edge of research within their fields, and have little time to spare to extensively search for practitioners who would be potential users of their results. While researchers often base their studies on practitioner problems, and may be involved in pilot studies with practitioners, they are generally limited to their small circle of immediate contacts. Thus we are well-placed to recognize fruitful connections that would otherwise go overlooked.

4. BROKERING COLLABORATIONS

On some occasions the connections we identified have been the springboard for immediate adoption of research techniques by practitioners. More commonly, however, merely making the connection is insufficient. Barriers remain that impede the adoption of a research technique. On the software practitioner side, the technique is often insufficiently mature to be a guaranteed match with their needs. In other words, practitioners should not, and will not, assume successful use of the technique as part of their critical development path. Furthermore, they are reluctant to devote their (very limited) time and effort to *trying* the technique. On the researcher side, typical research grants will cover the research itself of course, and perhaps a pilot study of its application (usually performed by the researchers themselves on representative data). However, they stop short of funding further maturation of the technique that would be more indicative of its usability (e.g., case studies where someone other than the researchers themselves apply the technique) and that would prepare it for third-party use (e.g., a well-rounded user interface, training material). To address these concerns our approach has utilized a pool of funding allocated specifically to support deployment of research techniques on projects. A primary goal is that a successful funded collaboration will lead to adoption of the technique by the software development organization.

Practitioner-led funding proposals

The research infusion team conducts an annual call for research collaboration proposals, distributing word of this through the channels discussed in section 4.

Proposals for such funding must be submitted by a software practitioner (not the technology provider), and must be for application of the technique to actual project use (not for further research).

Unlike other research programs, Research Infusion optimizes the likelihood of a successful collaboration by communicating with each proposal team (wherever possible) prior to the proposal due date to ensure, initially, that there is a good match of technique and requirements, that the proposed collaboration is well-designed, and finally that the nominal outcome of the project will be a success by our standards (see "Success Criteria and Progress Metrics" below).

Funding level

The funding range for each collaboration is $20,000 – $50,000 over a 6-month period. Funds are intended to be used for risk-reduction in introducing the technology—for example, for training, customer support, limited licenses where required, and collaboration management, data collection and analysis. Despite the low level of funding in comparison to typical NASA project budgets, we have seen an increase in the number of proposals over the three years in which Research Infusion has held competitions.

Proposal Selection criteria & process

Research Infusion established the following evaluation criteria for submitted proposals. The proposal template includes sections crafted to gather information on each of these criteria:

a) Feasibility: Is the proposed collaboration feasible? Are the skills of the participants relevant, the funding adequate, the management plan sound?
b) Impact on NASA: What will be the impact on NASA? Is the technique being applied to an important project?
c) Likelihood that, if successful, the technique will be adopted as part of the development team's practice: What is the likelihood that the technique, if successful in the proposed collaboration, will be adopted as part of the development team's practice?
d) Adequate feedback provided to researchers: Is adequate feedback provided to the researchers during the collaboration? For example, bugs, metrics data, final report.
e) Good use of NASA funds: Is the proposed collaboration a good use of NASA funds? The proposal's budget section addresses this question directly by stating how the funds will be used. We also ask that the proposer indicate what the impact will be on the development project if the proposal is *not* implemented.

When a collaboration proposal is received, each member of the Research Infusion team individually evaluates the proposal on each criterion (1 – 5 points for each criterion) and provided comments. These evaluations are then reviewed in a team meeting. In contrast to common proposal evaluation process, the team develops questions for the proposal teams and contacts them to obtain informal clarifications or even proposal revisions. The research infusion team's purpose in the extended communication is to enhance the proposed collaborations' prospects for success. The final group ranking, recommended funding level, and rationale is provided to the Software Assurance Research Program, which makes the final funding decisions.

Collaboration management

Following awards, we oversee the collaborations to ensure that practitioners and researchers are communicating, planning, and working toward their goals, keeping in mind the success criteria, and to report to the Research Infusion team lead the project status and particularly any issues, as they arise, that threaten success. Oversight requires facilitation of communication and feedback to both practitioners and researchers. This includes obtaining the researchers' perspective on the collaboration team's performance. The oversight team is familiar with other applications of the same or similar research, and has experience in evaluating software engineering research and its applications. The oversight team ensures collaboration start-up—transfer of funds, project

planning, training, etc.; evaluates and advises on experiment design and identifies other NASA sources for assistance for the collaboration – for example, individuals who have some experience with the technique; advises on defining collaboration-specific success criteria as well as the overall research Infusion success criteria; helps track success criteria.

Success Criteria and Progress Metrics

Our primary success criterion is that the research products used in the collaborations are adopted for future software development by the teams (or organization). However, this is unrealistic for mid TRL-level research products that may lack productization, and it may be unrealistic for high TRL or even for commercial products (for example, the license fee may be too high for a single team to bear). Thus we have identified several complementary success criteria:

a. The success criteria of the collaboration projects funded under this proposal are met. This includes a positive rating for each product on the collaboration's evaluation criteria metric(s).

b. The research product is adopted by the collaborating software development team for current use.

c. The research product is included in a list of recommended development practices at a NASA Center or by contractor.

d. The software development team using the product provides feedback, including performance data, to the research team to guide future development of the product.

e. Six months after the funded collaboration period, the research product is still being used by the development project or by a successor development project.

f. Independent of the success of the collaborations, "lessons learned" regarding the challenges and success factors for software development technology infusion within NASA.

Determination of return on investment for the technology within the scope of the collaboration is conspicuous by its absence from our evaluation criteria. This is partly because of the difficulty of determining ROI, especially given the extremely limited funding available to each collaboration. In addition, each software development organization uses its own procedure for determining whether to adopt a new technology; in some cases the procedure may involve an ROI determination, in other cases not. Our charter is to infuse the technology; success criteria such as whether the technology is adopted (item b) are direct measures of our success in achieving this goal, while ROI may be one factor in an organization's decision to adopt.

5. COLLABORATIONS 2004 - 2005

Our effort was chartered in 2002. We held NASA-wide videoconferences in August of 2003, May of 2004 and March 2005. At each of these we featured seven or more promising assurance techniques (in the second and third events, repeating some of the ones from previous years as well as new ones), and announced a "call for collaboration proposals". Following the selection process described earlier, this lead to funding for a selection of Research Infusion collaborations.

Ten such collaborations were initiated during 2004 – 2005. The technologies included a technique for conducting more efficient formal inspections; software defect classification for process improvement; requirements analyzers; code analyzers; and tools and a method for design rationale capture. The target application projects included spacecraft flight software, a ground antenna controller, International Space Station payloads, Space Shuttle and Space Shuttle Main Engine software, and a mission design activity. An additional four collaborations have been approved for 2006.

To date, six collaborations have completed, all of them achieving a "penetration factor" of 9 (as measured on the NASA Software Assurance Research Program's scale of 1 – 9)—the results of applying the technology were actually used on the project. In the historical context, this level of penetration of new software engineering technologies is rare. One collaboration resulted in success criteria (e) – technology is still in use 6 months after the end of the collaboration – and (c) – the technology is in the center's list of recommended development practices; two other collaborations are planning to adopt (and so would lead to (e)); and yet two more are investigating adoption in their context.

6. EXTRACTING LESSONS LEARNED

Lessons learned address questions such as: What additional guidance can collaborators be given to improve their success rate in the future? Why is technology transition difficult within NASA? What are the success factors for a research product to be adopted? What communication channels between researchers and practitioners within NASA can improve adoption?

In the remainder of this section we report some lessons learned based on the initial sets of Research Infusion collaborations.

Some developers are not proficient at research-oriented activities and need guidance and oversight. These teams are likely to benefit from more detailed *pro forma* documentation or templates (kick-off meeting agenda, project plan, final report). For specific categories of tools (such as static analysis tools) we can provide very detailed templates. They also require frequent oversight (a) to be sure communication is occurring between developers and researchers, and (b) to

verify that the schedule is being followed. Not all the projects require this level of support but such support is likely to benefit Research Infusion by promoting uniform, higher-quality collaboration practices.

Research Infusion's technology selection criteria have remained largely unaltered through several years of scrutiny an application. However, several modifications are recommended for the future.

A greater emphasis should be placed on the criterion "How easily can the research product(s) be integrated into a software development project?" While this is stated as a constraint on the technology, it is a relation between the technology and the development environment, and it requires more careful evaluation by the collaboration team prior to proposal submission. For example, several collaborations have had unexpected difficulty due to incompatibilities in the compiler (or other development tool) used on the project and the requirements of the technology. This can be a more serious issue at NASA than elsewhere because of the very conservative nature of NASA software development, supporting long-obsolete development platforms, in contrast to the most current environments that are typically supported by new software engineering technologies.

Also, the evaluation criteria for collaboration proposals need to take into account contractual risks (this has not been made explicit to the collaboration team to date). The question can be interpreted in "cost/benefit" terms—will so much time be spent on handling contractual issues that the collaboration is put at risk. Again, this is a particularly significant issue for NASA projects where there can be a high administrative overhead (including long delays as well as personnel effort) in getting necessary approvals. These obstacles have the potential for derailing projects with low funding and short duration.

Another risk that should be recognized and mitigated results from the classification of the collaboration's target software. Software that is classified as export-controlled may limit collaboration participation by technology developers. Unfortunately, the most safety- and mission-critical code is often classified as ITAR at NASA.

There are various answers to the question "What is the next step" – from research infusion to technology transfer. A general solution is unlikely. Some technologies are readily integrated and generalized into a parent organization's existing processes – they are modifications to existing processes. Various other technology-specific approaches may be appropriate within the NASA context.

Tighter qualification of technology / project combination may be needed. One of the static analysis tools used had previously been successfully applied to NASA software, but that software had specific technical features. The tool did not transition well to software that did not have these features. Also, the appropriate lifecycle context and purpose for the tool (in this case) may not have been clear to the development teams.

Collaborations' project plans should explicitly include an iterative approach to technology application, scaling up with each iteration.

To succeed, training and continued support are needed. For example, one of the static analysis tools lacked training, and minimal support was provided. The technology vendor did not visit the development team to train and consult on the tool's application. In contrast, another development team received onsite training on applying the technology it selected to its own application. This reduced risk and cost as well, since part of the target application was used in the training session. "The most successful way to do tech transfer is to put a member of the [technology vendor team] on the development team" – Matt Barry, JPL, (paraphrased) communication to the authors.

Overall, Research Infusion's first set of completed collaborations supports the hypothesis that with selection of appropriate technologies, careful matching of technology with software development team, and guidance and oversight, infusion of new software engineering technologies can be performed successfully on a minimal budget.

7. Discussion and Conclusions

Research Infusion has demonstrated an inexpensive and effective process for brokering matches between software engineering researchers and practitioners that can be incorporated into NASA's overall strategies for infusion of software engineering research products, and specifically for research products that can improve software safety and mission assurance.

As our procedures are codified and the research infusion team has gained experience, our approach is likely to scale to a greater range of software engineering technologies (not just those addressing software assurance) and to larger numbers of collaborations. Expansion of scope to more "revolutionary" technologies—technologies requiring a more significant change to an existing software development process model, or to the required infrastructure—is likely to require adaptations in the Research Infusion business model.

Acknowledgements

The research described in this paper was carried out at NASA Ames Research Center, Langley Research Center, Marshall Space Flight Center, and Goddard Space Flight Center, and at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. We thank Burton Sigal for his feedback on earlier drafts.

We wish to acknowledge the contributions of the following individuals and institutions. John Kelly, in the NASA Office of the Chief Engineer, leads the NASA intercenter Software Working Group and has provided support for the research infusion effort as part of the NASA Software Engineering Initiative. This Initiative was the basis for a software engineering research infusion effort. Pat Schuler of NASA Langley suggested our basic approach. Martha Wetherholt, in the NASA Office of Mission Assurance, is head of the NASA Software Assurance Research Program, which is administered by Kenneth McGill at the NASA IV&V Facility. Both have helped our infusion effort by providing collaboration funding, direction, and advice. Tim Menzies, formerly of West Virginia University and the IV&V Facility, now at Portland State University, was an early member of the team and helped give us our start. Wes Deadrick of the IV&V facility has also advised us and has been a reviewer of collaboration proposals. The Research Infusion team also wishes to acknowledge the many researchers who have lent their support and the software developers who have submitted collaboration proposals.

REFERENCES

[1] E. Rogers, *Diffusion of Innovation*, The Free Press, New York, 1983.

[2] A.A. Shapiro. "Technology Infusion for Space-Flight Program," *2004 IEEE Aerospace Conference Proc.*, Volume 1 Pages 662-667, March 6-13, 2004.

[3] M.V. Zelkowitz. "Software engineering technology infusion within NASA", *IEEE Transactions on Engineering Management*, 43(3) , pp. 250-261, August 1996.

[4] NASA Software Working Group web site http://software.nasa.gov/about/index.cfm

[5] Orr, James. "Space Shuttle Software Development and Certification", presentation at NASA Ames Research Center by United Space Alliance, October 2000.

[6] NASA headquarters web site http://www.hq.nasa.gov/office/codeq/software/

[7] NASA Independent Verification and Validation Facility's Office of Safety and Mission Assurance Software Engineering Research Program web site http://www.ivv.nasa.gov/forresearchers/osmasarp/osmasarp.php

[8] NASA Ames Research Center's Robust Software Engineering Group web site http://ti.arc.nasa.gov/ase/index.html

[9] Jet Propulsion Laboratories' Laboratory for Reliable Software web site http://eis.jpl.nasa.gov/lars/

[10] NASA Langley Research Center's Formal Methods Group web page http://shemesh.larc.nasa.gov/fm/

[11] NASA Goddard Space Flight Center's Software Engineering Laboratory web site http://sel.gsfc.nasa.gov

[12] J.C. Mankins. "Technology Readiness Levels", White Paper, Advanced Concepts Office, Office of Space Access and Technology, NASA, April 1995, available from http://www.hq.nasa.gov/office/codeq/trl/trl.pdf

[13] J.D. Smith II. "An Alternative to Technology Readiness Levels for Non-Developmental Item (NDI) Software", *Proc. 38th Hawaii International Conference on System Sciences*, 2005.

[14] R. Mackey, R. Some & A. Aljabri. "Readiness Levels for Spacecraft Information Technologies", 2003 IEEE Aerospace Conference Proceedings, Volume 1 Pages 391- 398, March 8-15, 2003.

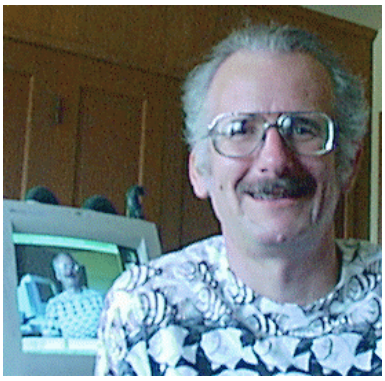
[15] NASA Software Engineering Research Infusion web site http://ti.arc.nasa.gov/researchinfusion

BIOGRAPHY

Tom Pressburger *is in the Robust Software Engineering group led by Dr. Michael Lowry at NASA Ames. He serves on the Ames Engineering Process group and is the alternate Ames representative to the Software Working Group. He led the software engineering research infusion subgroup described in this paper. Lately, he has been working on projects related to reuse of software for NASA's Exploration mission. His expertise is in the area of program synthesis which was applied to Java model checking, state estimation, statistical algorithms, solar system geometry, and, when he was at the Kestrel Institute, algorithm design. He also worked at Reasoning Systems in the area of software reengineering. He holds a B.S. in Mathematics from CalTech and an M.S. in Computer Science from Stanford.*

Ben Di Vito *is a senior research engineer in the Safety-Critical Avionics Systems Branch at NASA Langley Research Center, where he applies formal methods to problems in fault-tolerant computing and flight-critical avionics. He has extensive experience with deduction tools and techniques, especially the PVS theorem prover for higher order logic. A recent research project initiated by Ben concerns the establishment of a mathematical database service called Hypatheon. Ben holds a Ph.D. in computer science from the University of Texas at Austin*

Michael Hinchey *is Director of the NASA Software Engineering Laboratory, located at Goddard Space Flight Center. He has held academic positions at the level of Full Professor in the USA, UK, Ireland, Sweden and Australia. He is the author of move than 200 technical papers, and 15 books. His current research interests are in the areas of formal methods, system correctness, and agent-based technologies. Dr. Hinchey is a Senior Member of the IEEE, a Fellow of the IEE and the British Computer Society. He is currently Chair of the IEEE Technical Committee on Complexity in Computing, and is the IEEE Computer Society's voting representative to IFIP TC1 for which he has been elected Chair for 2006 to 2008. He received the Ph.D. in Computer Science from University of Cambridge.*



Martin S. Feather *is a Principal in the Software Quality Assurance group at JPL. He works on developing research ideas and maturing them into practice, with particular interests in the areas of early phase requirements engineering and risk management and of software validation (analysis, test automation, V&V techniques). He obtained his BA and MA degrees in mathematics and computer science from Cambridge University, England, and his PhD degree in artificial intelligence from the University of Edinburgh, Scotland. For further details, see http://eis.jpl.nasa.gov/~mfeather*

Lawrence Markosian *is a Computer Scientist with QSS Group, Inc. at NASA Ames Research Center, where he led a team developing model checking tools based on Java PathFinder. He is a member of the NASA Software Engineering Initiative's Research Infusion team. Prior to joining NASA, he was a founder of Reasoning Systems., where as VP of Applications Development he managed technology transfer of advanced software engineering tools. Markosian has an undergraduate degree in mathematics from Brown University and has done graduate work at Stanford University in logic and artificial intelligence.*

Luis Trevino *is Chief Scientist and contributing partner with 2L Research Corporation, designing and developing advanced software algorithms for DoD. Prior to joining 2L Research, he was involved with cutting edge NASA programs for over 16 Years. He has worked and led projects in the Advanced Sensors & Health Management Systems Branch, Spacecraft & Vehicle Systems Department of the Engineering Directorate at Marshall Space Flight Center. He received his BSEE from Texas A&M University and his MS and Ph.D. in Electrical Engineering from the University of Alabama in Huntsville.*