Report from Dagstuhl Seminar 22052

The Human Factors Impact of Programming Error Messages

Brett A. Becker[*1], **Paul Denny**[*2], **Janet Siegmund**[*3], **Andreas Stefik**[*4], **and Eddie Antonio Santos**[†5]

1 **University College Dublin, IE.** brett.becker@ucd.ie
2 **University of Auckland, NZ.** p.denny@auckland.ac.nz
3 **TU Chemnitz, DE.** siegj@hrz.tu-chemnitz.de
4 **University of Nevada - Las Vegas, US.** stefika@gmail.com
5 **University College Dublin, IE.** eddie.santos@ucdconnect.ie

Abstract

The impacts of many human factors on how people program are poorly understood and present significant challenges for work on improving programmer productivity and effective techniques for teaching and learning programming. Programming error messages are one factor that is particularly problematic, with a documented history of evidence dating back over 50 years. Such messages, commonly called compiler error messages, present difficulties for programmers with diverse demographic backgrounds. It is generally agreed that these messages could be more effective for all users, making this an obvious and high-impact area to target for improving programming outcomes. This report documents the program and the outputs of Dagstuhl Seminar 22052, "The Human Factors Impact of Programming Error Messages", which explores this problem. In total, 11 on-site participants and 17 remote participants engaged in intensive collaboration during the seminar, including discussing past and current research, identifying gaps, and developing ways to move forward collaboratively to address these challenges.

Seminar January 30–February 02, 2022 – http://www.dagstuhl.de/22052

2012 ACM Subject Classification General and reference → Surveys and overviews; Human-centered computing → Human computer interaction (HCI); Social and professional topics → Computing education; Applied computing → Education; Software and its engineering → General programming languages

Keywords and phrases Accessibility; Compilers; Compiler Error Messages; Computer Science Education; Computer-Human Interaction; Computing Education; CS1; Error Messages; HCI; Human Factors; Human Computer Interaction; Novice Programmers; Programming Errors; Programming Error Messages; Programming Languages; Software Engineering; Syntax Errors

Digital Object Identifier 10.4230/DagRep.12.1.119

[*] Editor / Organizer
[†] Editorial Assistant / Collector

1 Executive Summary

Brett A. Becker (University College Dublin, IE, brett.becker@ucd.ie)
Paul Denny (University of Auckland, NZ, p.denny@auckland.ac.nz)
Janet Siegmund (TU Chemnitz, DE, siegj@hrz.tu-chemnitz.de)
Andreas Stefik (University of Nevada - Las Vegas, US, stefika@gmail.com)

Programming error messages (commonly called compiler error messages) pose challenges to programmers – from novices to professionals – with evidence dating from the 1960s to present day. In this seminar, we explored the nature of these challenges and particularly why they remain largely unaddressed. We further investigated the specific challenges that different users including children, non-native English speakers, and those of varying ability experience when faced with programming error messages. Finally, we sought to identify the most promising avenues to assess the effectiveness of error messages, how to improve them with large, demonstrated effect, and how to produce appropriate messages for different users with different needs. To this end, we assembled experts from many sub-disciplines of Computer Science, including Programming Languages, HCI, Computer Science Education, and Software Engineering as well as Learning Sciences. Due to travel restrictions imposed by the COVID-19 pandemic, we ran the seminar in a hybrid format, with 11 on-site participants collaborating with 17 remote participants. We formulated a schedule for the seminar that sought to maximise outcomes for all participants.

By combining the expertise of these different disciplines, we could identify gaps in knowledge and high-priority areas to build a basis for future work. This is the starting point for a long-lasting contribution to the field. By uniting communities that have to date been working largely in isolation, we sought to gather appropriate and useful data, broaden perspectives, build consensus among diverse stakeholders, and begin cross-community efforts working in unison going forward.

During the seminar, we had sessions focusing on existing literature and practice, including experience reports from language maintainers, expert users, researchers, and educators, to develop a shared understanding of the evidence that exists with regard to effective programming error messages. This included the group working together to synthesise existing data sets, which we view as an important exercise given that large corpora of errors and error message data are generally not openly available. This can make it difficult for researchers to answer seemingly simple questions such as: “What are the most frequent error messages encountered by [students, professionals, blind, non-native English speakers] in language x?” or, “What are evidence-based examples of effective and/or ineffective error messages?”. As a result, we identified several fruitful avenues in the form of cross-discipline collaboration, data sharing opportunities, and improved focus on what steps are needed to improve efforts to answer these questions.

One of the key objectives of the seminar was to identify areas for immediate research. Several problems have been identified, including but not limited to:

1. linking error messages with the context of the problems and programs being worked on when error messages are generated;
2. understanding the metacognitive aspects involved in the process of interpreting and reacting to error messages;
3. identifying what error messages arise from inconsistent conceptions of how a program runs or how it is structured;

4. the effects of error message presentation on error message interpretation (e.g. visual queues, structured error messages, etc.);
5. error message classification schemes;
6. metrics to measure and assess error messages;
7. identifying when error messages are "wrong", what occurs when they steer the programmer in the wrong direction, and how this can be avoided;
8. determining the design factors of programming error messages that differ across demographic groups (e.g., expertise level, disabilities, native/natural language, etc.).

Another key objective was to establish new cross-community efforts to improve programming error messages in practice, leveraging the strengths of each community. Participants discussed open research problems and identified those of high priority and interest including those listed above. Several interdisciplinary research objectives have been established, and seeds were sewn to form teams to collaboratively address these research questions.

2 Table of Contents

3 Overview of Talks

3.1 After ++5 Decades of Error Messages, Where Do We Go (right) Now?

Brett A. Becker (University College Dublin, IE)

Joint work of Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, James Prather
Main reference Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, James Prather: "Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research", in Proc. of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE 2019, Aberdeen, Scotland Uk, July 15-17, 2019, pp. 177–210, ACM, 2019.
URL http://dx.doi.org/10.1145/3344429.3372508

Text-based error messages are the primary mechanism provided to programmers by a programming language to fix errors in code. Error messages have been studied for over 50 years in the context of education, software engineering, human-computer interaction, and programming languages. They are often described as confusing and misleading, and are known to cause frustration, particularly for students and novices, but also for professionals. They are situated in the complex boundary between the human user and the system and involve constructed computer languages, human language, cognition, and complex concepts such as the notional machine. They have impacts on humans and society, including the economy, the workplace, and human and societal productivity. They also likely impact diversity, equality, and inclusion as they affect various people and groups differently. The historical literature has called for several avenues to be explored in order to improve their effectiveness. Although progress has been made on this front in recent years many areas remain largely under-explored. New advancements in cognitive science and neuroscience, as well as approaches based on artificial intelligence show promise in improving error messages. However much fundamental work remains. This presentation provides an overview of this landscape, presents a corpus of 330 scientific papers on programming error messages spanning 1965–2019 [1], and provides several avenues for future work – from fundamental human factors research to those based on emerging technologies.

References

1 Becker, Brett A., & Denny, Paul, & Pettit, & Bouchard, Durell & Bouvier, Dennis J. & Harrington, Brian & Kamil, Amir & Karkare, Amey & McDonald, Chris & Osera, Peter-Michael & Pearce, Janice L. & Prather, James, *Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research*. In Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19), Association for Computing Machinery, New York, New York USA, 2019, `https://doi.org/10.1145/3344429.3372508`

3.2 Teaching Novices to Read (gcc) Programming Error Messages?

Dennis Bouvier (Southern Illinois Univ. Edwardsville, US)

Until better tools are commonly used, students could benefit from being better prepared to make use of error messages in current systems. Presented are: (1) Ideas for essential topics on for novice programmers learning to use error messages effectively. The topics include (1a) terminology, (1b) compiler technology for producing error messages, and (1c) strategies for using error messages effectively. (2) Ideas for evaluating the efficacy of the lesson, including (2a) debugging activity, and (2b) a questionnaire of personal debugging practice. In the debugging activity, students are presented with a sequence of programs that each have one error message. The student then (i) compiles the program. (ii) reports the error message found, (iii) explains the error message in their own words, and (iv) proposes a fix for the code without actually editing the code nor recompiling. The questionnaire of personal debugging practice asks the student to report on activities they employ in response to error messages. These activities were experienced by students as the third lesson of an introduction to programming course using the C programming language. The debugging activity and questionnaire were repeated two-thirds through the course. Data and analysis to be reported at a later date.

3.3 Blackbox

Neil Brown (King's College London, GB)

Blackbox is a large dataset of programming activity, collected worldwide from users of the BlueJ IDE that helps novices learn to program Java. In this session I will describe the dataset and then guide attendees through an activity to explore some of the data and see how novices respond to error messages "in the wild". Finally, I will describe Blackbox Mini, a new small subset of the data with a simple schema.

3.4 Improving Programming Error Messages: Why Should We Bother?

Paul Denny (University of Auckland, NZ)

It is well known that programming error messages can be notoriously difficult for novices to understand. This is a long standing problem and one that, if solved, would have clear positive impacts on student learning. At this seminar, all participants bring with them a wonderful diversity of motivations, interests and expertise for tackling this problem. I will articulate my own motivations, which include that students have suffered long enough, and will support these with empirical data illustrating the large positive effects that better error message have on the programming performance of novices.

3.5 Error Messages: Six Seconds of Progress from 25 Years of Research

Kathi Fisler (Brown University, Providence, US) and Shriram Krishnamurthi (Brown University, Providence, US)

We have made significant progress in our theoretical and practical understanding of error messages. Some of this work is in languages and environments outside the mainstream; in other cases, these are general results that can be applied widely.

The talk presents eight vignettes describing results, challenges, and cautions for interpretation.

3.6 Observing Programmers with EEG and Eye Tracking

Janet Siegmund

To gain deeper insights into how programming error messages are perceived by programmers, we can use established methods from neuroscience and cognitive psychology, especially electroencephalography and eye tracking. We demonstrate a possible experiment with EEG and eye tracking and show what we could do with the data.

3.7 On the Location of Errors in Source Code

Tobias Kohn (University of Cambridge, GB)

Compilers tend to present errors at specific locations in the source code, indicating where exactly the parsing process or name resolution ground to a halt. We might assume that this should also be the region to make corrections and amend the source code. Closer inspection, however, reveals that a substantial part of the errors are not that easily pinned to a specific location. In fact, it might be more fruitful to think of compiler errors primarily as *inconsistencies* in the source code. Consider a variable name, for instance, that is spelled differently – who is to say that the variable's declaration must be the correct spelling with all other deviating occurrences being wrong?

On the basis of several examples of student code we make a case that indeed most programming errors may be better viewed as matters of inconsistencies in source code rather than clear-cut mistakes. Presenting programming errors as such inconsistencies might also work towards the learners understanding as to why the computer has trouble "understanding" the program code, with less emphasis on arbitrary syntactic rules that must be followed.

3.8 Novices vs Expert Errors: One Size Fits None?

Linda McIver (ADSEI, Glen Waverley, AU)

A lively discussion of what we need from error messages, and whether novice and expert programmers' needs are irreconcilably different. The consensus, if there was one, seemed to be that novices and experts require different things, but there were a few assumptions that require further testing.

3.9 Error Message Topic Potpourri

Prajish Prasad (IIT Bombay, IN)

In this session, I facilitated a discussion where we discussed three topics related to error messages. For each topic, I provided some background and focus questions. Discussions were centred around these focus questions.

The first topic was related to "The Effect of Error Messages in Learners' Metacognition and Self-Regulation". Metacognition, in simple words is "thinking about one's own thinking". When learners encounter a programming task, applying metacognition and self-regulation strategies involve identifying strategies that have been successful or unsuccessful in the past and evaluating these strategies based on feedback. We discussed how error messages can hinder learners' metacognitive processes and whether we could provide general metacognitive scaffolds to help learners recover from these errors.

The second topic was related to "Error Messages and Live Coding". Live coding is the process of writing code live on a computer. In recent times, several developers as well as instructors have been doing live coding. We discussed whether general patterns or guidelines can be extracted from live coding videos that show expert developers encountering and fixing error messages.

Finally, we discussed about "Error Messages in Web Programming Languages", and the challenges in designing and improving error messages for such languages. The web has so many languages and implementations, that deciphering errors often requires a strong understanding of the architecture of the web, of browsers, and of languages.

3.10 Terminology Used When Discussing Programming Error Messages

Eddie Antonio Santos (University College Dublin, IE)

I was interested in the seminar participants' views and attitudes about the words we use when talking about programming error messages (PEMs). I divided this talk into two sections:

1. I had participants post their favourite error message. "Favourite" was defined however the submitter chose.

2. I had participants collectively edit a Google Document adding to a list of salient words found within programming error messages: and a list of words that us – educators, language implementers, and researchers – use when talking *about* PEMs.

Our discussion revealed that our attitudes towards the wording found within contemporary PEMs is largely negative, and can be improved in terms of tone. We note that for experts, words like “illegal”, “offending”, “fatal”, are okay, however novices may be negatively affected by these “scary” words. We suggest that compiler developers may wish to add another stage to the compiler explicitly dedicated to providing feedback to the programmer. This module should be maintained by a professional in human-computer interaction. We also float the idea that “error” is the wrong word to use when discussion feedback whatsoever.

4 Panel discussions

4.1 inHUMANe Programming Error Messages – Position Panel

Dennis Bouvier (Southern Illinois Univ. Edwardsville, US), Kerstin Haring (University of Denver, US), Felienne Hermans (Leiden University, NL), Amy Ko (University of Washington, Seattle, US), Michael Kölling (King’s College London, GB), and James Prather (Abilene Christian University, US)

The organisers of the workshop posed position statements to the panel. Panellists discussed both sides of several positions, including (1) Error messages for novice programmers should be avoided at all costs, (2) Kids should learn programming before learning to read and write natural languages, and (3) Internationalised programming error messages are harmful.

With respect to position 1, the position was supported and opposed by most panellists. For example, the argument can be made that error message understanding is crucial for forming professional skills and/or for building knowledge of how programming systems work; likewise the argument can be made when the goal is building confidence with computing, or using computing as a means to study another discipline, learning should not be hampered by dealing with avoidable error messages.

With respect to position 2, the position is supported by some panellists considering that programming is (or will be) a fundamental skill, and that learning programming early in life could be beneficial. Most panellists pushed back on the use of the word “before” in the position, stating that learning to code could, and perhaps should, coincide with learning natural language(s).

With respect to position 3, multiple panellists posed the possibility of using programming error messages expressed in a natural language being learned as a way to expand the programmer’s experience in the language while programming; though, it was not entirely clear that such suggestions were entirely genuine. Also suggested was that American programmers would not be able to adapt to messages in a foreign language, and so all error messages should be in English. As this round progressed, it became clearer that, at least, half of the panellists’ positions were somewhat facetious. Taking a tangent in the discussion, panellists suggested that no written language be used – errors could be conveyed by “screaming” and/or “electric shock” or memes.

The final point of discussion was "how is, or how can, change of programming technology be influenced for the better?" Michael Kölling suggested that "creating new better things" is the way to influence the way forward, and gave Snap*!* as an example. Amy Ko suggested that organised community effort can be a way forward, and that influencing "error messages" is a unique opportunity for influencing. Ultimately, the panel suggested this was an Hegelian dialectic; and that making error messages more understandable might fit some goals, but not others.

4.2 Directions for Error Message Research

Jan Pearce (Berea College, US), Neil Brown (King's College London, GB), Linda McIver (ADSEI, Glen Waverley, AU), Jens Mönig (SAP SE, Walldorf, DE), and Raymond Pettit (University of Virginia, Charlottesville, US)

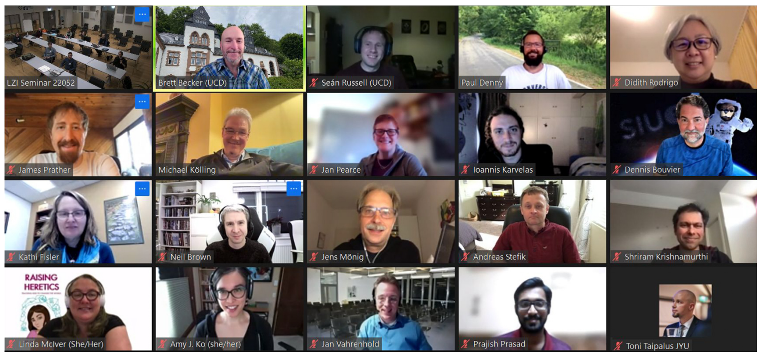
Neil Brown, Linda McIver, Jens Mönig, Jan Pearce, and Raymond Pettit served as discussants, presenting their views on the impact of error messages on identity and self-efficacy, the needs of novice programmers versus expert programmers, what research is needed in these key areas, as well as what concerns panellists have as reviewers of conference papers on error messages. Panellists also discussed how one might make the delivery of error messages more nuanced as well as more accessible to those with disabilities. Neil Brown talked about activity traces from the Blackbox data, and how error messages can often send students down the wrong route to fix their code, thus increasing their frustration and damaging their confidence. Linda McIver talked about how many students come to programming convinced that it's too hard, and that unintelligible error messages are seen as proof. These students are looking for evidence that they can't program, and bad error messages provide that proof. Jens Mönig discussed his concern that expectations regarding the positive and negative impacts of error messages may be too large given that these impacts are so heavily dependent upon many other contextual variables that are difficult to disambiguate. Jan Pearce shared and led a discussion on student-produced artefacts that self-describe the negative impacts error messages have personally had on them. Participants discussed whether these self-described negative impacts were due to the error message content or the error itself. Raymond Pettit discussed the differences in the ways novices and experts relate to error messages. Experts are accustomed to receiving these messages and use them as a tool in helping to fix a program, whereas novices often see the error message as the problem. Attendees commented on the ideas shared.

Participants

- Annabelle Bergum
Universität des Saarlandes – Saarbrücken, DE
- Joe Dillane
University College Dublin, IE
- Kerstin Haring
University of Denver, US
- Elisa Madeleine Hartmann
TU Chemnitz, DE
- Felienne Hermans
Leiden University, NL
- Tobias Kohn
University of Cambridge, GB
- Jens Mönig
SAP SE – Walldorf, DE
- Raymond Pettit
University of Virginia – Charlottesville, US
- Ma. Mercedes T. Rodrigo
Ateneo de Manila University – Quezon City, PH
- Eddie Antonio Santos
University College Dublin, IE
- Janet Siegmund
TU Chemnitz, DE

Remote Participants

- Brett A. Becker
University College Dublin, IE
- Dennis Bouvier
Southern Illinois Univ. Edwardsville, US
- Neil Brown
King's College London, GB
- Paul Denny
University of Auckland, NZ
- Kathi Fisler
Brown University – Providence, US
- Ioannis Karvelas
University College Dublin, IE
- Amy Ko
University of Washington – Seattle, US
- Michael Kölling
King's College London, GB
- Shriram Krishnamurthi
Brown University – Providence, US
- Linda McIver
ADSEI – Glen Waverley, AU
- Jan Pearce
Berea College, US
- Prajish Prasad
IIT Bombay, IN
- James Prather
Abilene Christian University, US
- Seán Russell
University College Dublin, IE
- Andreas Stefik
University of Nevada – Las Vegas, US
- Toni Taipalus
University of Jyväskylä, FI
- Jan Vahrenhold
Universität Münster, DE


LZI Seminar 22052
Brett Becker (UCD)
Seán Russell (UCD)
Paul Denny
Didith Rodrigo
James Prather
Michael Kölling
Jan Pearce
Ioannis Karvelas
Dennis Bouvier
Kathi Fisler
Neil Brown
Jens Mönig
Andreas Stefik
Shriram Krishnamurthi
Linda McIver (She/Her)
Amy J. Ko (she/her)
Jan Vahrenhold
Prajish Prasad
Toni Taipalus JYU