University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Compiler Error Message Enhancement

What’s been done, what needs doing

Brett A. Becker
brett.becker@ucd.ie
www.brettbecker.com

Dagstuhl Seminar 18061, Evidence About Programmers for Programming Language Design, February 4 - 9, 2018

SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

Overview

Compiler Error Messages

Problems and importance

Recent Results

Motivating evidence for enhancing compiler error messages
Empirical evidence of effects

Where Are We Now?

Metrics and signals
Compilers and languages
Frameworks, guidelines, principles, no-nos, rubrics, etc.

Moving Forward

Where do we go from here?

This talk is based on several papers by the author, and the related literature. The interested reader is guided to the following paper for a good entry point into the literature on compiler error messages:

(Becker *et al.*, 2016): Becker, B.A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K. and Mooney, C. Effective Compiler Error Message Enhancement for Novice Programming Students, *Computer Science Education* 26(2-3), pp. 148-175, 2016
http://dx.doi.org/10.1080/08993408.2016.1225464

This and the author's other papers on the topic can be found at www.brettbecker.com/publications

Terminology

Unfortunately, deemed necessary

- long repeated phrases can be distracting (particularly on slides)
- specificiy is important, and laziness hard to avoid
    - ’compiler error messages’ turn in to ’compiler errors’ turn in to ’errors’, and we need that word for another thing:

Term	Meaning
CEM	Compiler Error Message
ECEM	Enhanced Compiler Error Message[1]
Error	error in code which causes CEM/ECEM

[1] Loosely, any error message that is based on a ’stock’ CEM but altered*, hopefully to make it ’better’ and/or a CEM that had some thought put into it from the user’s point of view. *Normally altered wording, but could include highlighting, etc.

Compiler error messaages: What's the problem?

CEMs...

- most often reflect the compiler's problem with the program (violation of specification) - not necessarily what the programmer did, and needs to fix
- pose particular difficulties for novices (but also affect pros)
    - are often misleading, or **wrong**, at least from the programmer's perspective
        - Marceau et al. (2011a,b) use the term counterfactual
- given the tight connection between error reporting and linguistic decisions such as parsing strategies, error message issues are clearly a Programming Languages problem as much as one of HCI (Marceau *et al.*, 2011b) as well as Education (me, today)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

A motivating example

Code:

```
class MyClass{
    void f(){
       int n = 10;
    // Error, closing brace is missing
    void g() {
       int m = 20;
    }
}
```

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

A motivating example

Compiler error message:[2]

```
C:\Users\brett\Desktop\junk\MyClass.java:5: error: illegal start of expression
        void g() {
        ^
C:\Users\brett\Desktop\junk\MyClass.java:5: error: ';' expected
        void g() {
              ^
C:\Users\brett\Desktop\junk\MyClass.java:8: error: reached end of file while parsing
    }
     ^
3 errors

Process Terminated ... there were problems.
```

[2] This was from javac, Java SE JDK 1.8.0_112. Ben-Ari (2007) got a very similar thing with the same code 11 years ago except the second error was reported on line 5 and the final error was '}' expected, on line 9.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

(some) issues with example CEM on previous slide

1. too verbose - unnecessary information
2. difficult to read - could it be laid out better?
3. counterfactual most of the line numbers seem wrong
4. accusatory - attributes blame to the user (more than anything else)?
5. negative phrasing - 'terminated', 'illegal'
6. vague (at least more than it could be)
7. (kind of) suggests a fix that is possibly not correct
8. redundant information
9. potentially spurious errors reported - are the second and subsequent reported errors due to genuine issues in the source, or due to a preceding reported error? Did I 'make' these errors happen? Should these even be here? If not, why are they?

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Diversity and Inclusion

Can compiler error messages affect diversity and inclusion?

- negative phrasing, assignment of blame to user, confusion, frustration
- I think yes. How a student reacts and internally deals with emotions invoked by the above shouldn’t be overlooked.
- I have experimented with trying to make *worse* CEMs than this one. It’s actually hard to achieve much worse than the previous example.
- CEM issues and presentation have great potential to affect those with disabilities differently.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Compiler error messages: What’s the problem?

Issues with CEMs have been well documented for over 50 years

inadequate and not understandable (Moulton and Muller, 1967), **useless** (Wexelblat, 1976), **not optimal** (Litecky and Davis, 1976), **inadequate (again, 17 years later)** (Brown, 1983), **frustrating** (Flowers *et al*., 2004), **cryptic and confusing** (Jadud, 2006), **notoriously obscure and terse** (Ben-Ari, 2007), **undecipherable** (Traver, 2010), **still very obviously less helpful than they could be** (McCall and Kölling, 2014), **inscrutable** (Ko, 2017), and **a source of frustration and a barrier to progress** (Becker *et al*., 2018b).

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Compiler error messaages: What's the problem?

Yet, ask any experienced programmer about the quality of error messages in their programming environments, and you will often get an embarrassed laugh. In every environment, a mature programmer can usually point to at least a handful of favourite bad error responses. When they find out that the same environment is being used by novices, their laugh often hardens (Marceau et al.*, 2011a, p. 499)*

Multiple compiler error messages

Multiple ‘cascading’ CEMs pose particular problems for novices.

- Becker *et al.* (2018b) analyzed 21 million Java CEMs from Blackbox (Brown *et al.*, 2014)
- frequencies of first CEMs were compared to all CEMs (including multiple ’cascading’ CEMs)
  - the frequencies of many CEM types change when considering first CEMs only
  - this indicates that researchers (who normally don’t look at first CEMs only) have an incorrect picture of what CEMs novices are often acting upon[3]
  - this may have important implications for teaching, tool design, and future research efforts

[3] e.g. many novice editors only show the first CEM to the user

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Compiler error messages: What's the problem?

- CEMs can pose issues for experienced developers for a range of reasons (Traver, 2010; Seo *et al.*, 2014)
- CEMs also cause issues for the educators who teach novices
    - very common for multiple students to have essentially identical issues with the same CEM - a notorious complaint of laboratory demonstrators and known cause of unproductively spent time
    - educators feel that they are seeing students make the same errors over and over (Thompson, 2004)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Important but problematic

The importance of CEMs (and novices' difficulties with them) can be anecdotally ascertained from a small sample of paper titles:

- **What the Compiler Should Tell the User** (Horning, 1974)
- **Error Messages: The Neglected Area of the Man/Machine Interface** (Brown, 1983)
- **Fatal Error in Pass Zero: How Not to Confuse Novices** (Du Boulay and Matthew, 1984)
- **The Neglected Battlefield of Syntax Errors** (Kummerfeld and Kay, 2003)
- **Compiler Error Messages: What Can Help Novices** (Nienaltowski *et al.*, 2008)
- **On Compiler Error Messages: What They Say and What They Mean** (Traver, 2010).

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Importance

- primary and instantaneous source of feedback, particularly in the absence of an instructor
- one of the most important tools and critical UX elements that languages offer to programmers (especially for novices) (Marceau *et al.*, 2011a; Traver, 2010)
- CEMs have some demonstrated utility in research, for instance the Error Quotient, a metric of how much a student is struggling (Jadud, 2006)
    - other metrics also utilize CEM information
- students' algorithms can change, possibly inadvertently, while fixing errors (Lane and VanLehn, 2005)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Importance

CEMs play at least two critical roles (Marceau *et al.*, 2011a)

- as a programming tool, they should help the user progress towards a working program
- as a pedagogic tool, they should help the user understand the problem that led to the error
- in addition, they must avoid frustrating[4] the user further

"As educators, we have a limited amount of time to spend with each student so providing students with automated and useful feedback about why they are getting syntax errors is very important" (Denny *et al.*, 2011, p. 278).

[4]this will come up a lot, later

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Importance

- arguably, the most important programmers are beginners
    - everyone was a beginner - tomorrow's programmers are today's beginners
- CS programs have BIG (and early) dropout issues, and programming is the number one suspect
- tools that make things hard probably affect diversity and inclusion
- ...

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Syntax itself is problematic too

what does that say about syntax errors?

Stefik and Siebert (2013)

- languages using a more traditional C-style syntax (both Perl and Java) did not afford accuracy rates significantly higher than a language with randomly generated keywords...
- ...but languages which deviate (Quorum, Python, and Ruby) did
- supports the broad conclusion of Denny *et al.* (2012): all syntax errors are not created equal

Last seven or so years

- prior to 2010 most evidence for ECEM effectiveness was anecdotal[5]
- 2010 on:
    - important empirical evidence motivating CEM and ECEM research
    - growing body of empirical work investigating effects of CEMs and ECEMs
    - increased interest in tools that provide ECEMs and languages that strive to provide useful CEMs (more later)
    - but... some apparently contradictory results on effectiveness
        - different metrics and methodologies make direct comparisons between studies difficult
- 2018: time for reflection?

[5] a few exceptions, many mentioned to some degree later, plus Harrington (1984) - see slide 'Academic archaeology' at end

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Barik *et al.* (2017)

eye-tracking study

Motivating evidence to improve CEMs (beyond historical gripes and the obvious)

- students allocate **12-25%** of their total task time to reading CEMs
  - Pettit *et al.* (2017) and Prather *et al.* (2017) also provide evidence that students read CEMs
- difficulty reading compiler error messages...
  - is comparable to the difficulty of reading source code
  - significantly predicts participants' task performance
  - contributes to the overall difficulty of resolving errors

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Lee and Ko (2011)

- hypothesis: attributing failure to the computer, rather than the learner, may improve learners' motivation to program
- two-condition controlled experiment manipulating a game's level of personification
- 116 self-described novice programmers recruited on Amazon's Mechanical Turk
- when given the option to quit at any time, those in the experimental condition (with a personable game character) completed significantly more levels in a similar amount of time
- findings suggest that how programming tool feedback is portrayed to learners can have a significant impact on motivation to program and learning success

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Marceau *et al.* (2011a,b)

- applied human-factors research methods to explore the effectiveness of ECEMs.
- languages: Racket/Scheme
- provide 'concrete' observations on effectiveness
- investigated specific error message features such as vocabulary, highlighting

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Denny *et al.* (2014)
setup

- Control/Treatment
- Editor/Language: Codewrite/Java
- Students: 83
- Submissions: approx 10,000
- Time: 2 weeks
- Particulars
  - Codewrite provides some skeleton code
  - Students used CodeWrite for exercises only
  - First two CEMs shown to control, but only first ECEM shown to treatment
  - ECEMs not shown side-by-side with CEMs

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Denny *et al.* (2014)

results

- **no** effect of ECEMs on the following metrics:
  - number of consecutive non-compiling **submissions**
  - total number of non-compiling **submissions**
  - number of **attempts** needed to resolve **three** most common CEM *categories*

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker (2015, 2016)

setup

- Control/Treatment
- Editor/Language: Decaf/Java
- Students: 200+
- Errors: approx 50,000
- Time: 4 weeks
- Particulars
  - No skeleton code
  - Students used Decaf for all programming (exercises, assessment, practice)
  - First CEM/ECEM only shown to students (as per BlueJ, etc.)
  - ECEMs shown side-by-side with CEMs (as per Coull and Duncan (2011))

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker (2015, 2016)

results

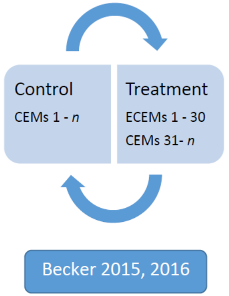
- **positive** effect of ECEMs on the following metrics:
  - total number of **errors**
  - number of **errors** per student
  - total number of repeated (consecutive) **errors**
- Identified eight specific Java CEMs with significantly reduced frequency when enhanced
- Student questionnaire responses also promising

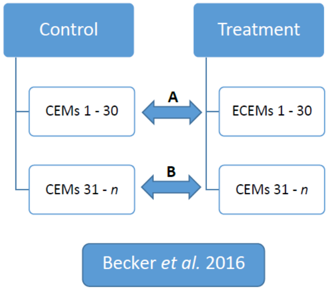
University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2016)

setup

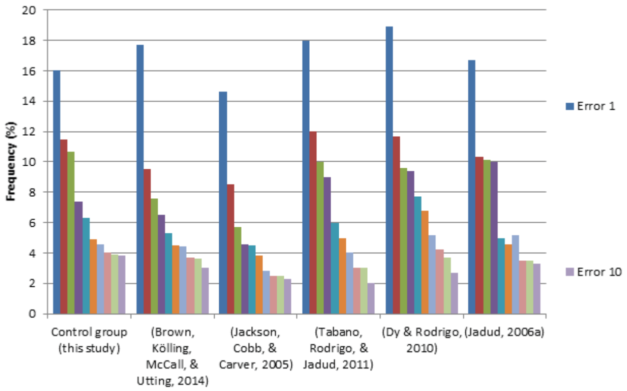
Similar data to Becker (2015, 2016), but different approach for analysis





University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2016)
control group sanity check



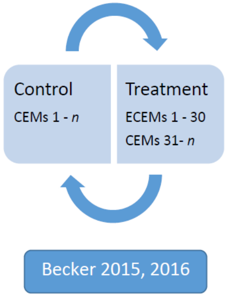
Percent frequency of the 10 most frequent Java CEMs (control group) and five other Java studies. In each group: left-most bar $\rightarrow$ most frequent CEM; right-most bar $\rightarrow$ 10th most frequent CEM.

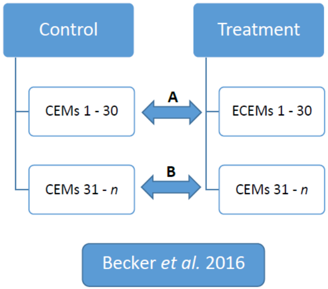
University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2016)

results

Similar data to Becker (2015, 2016), but different approach for analysis





Comparison A corroborated Becker (2015, 2016). Comparison B found no statistically significant difference.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2016)

results

CEM description
class, interface, or enum expected
not a statement
incompatible types
.class expected
cannot find symbol
{ expected
<identifier> expected
illegal start of expression
variable already defined

Nine CEMs for which enhancement showed statistically significant reduction of errors per student

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2016)

Notice that some very common yet simple CEMs did not show a reduction of errors per student.

- for instance, *; expected* is pretty straightforward
- corroborated by Harker (2017)
- if results showed that *; expected* was affected, it might be a sign of something else at work

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* and Denny *et al.* (2014)

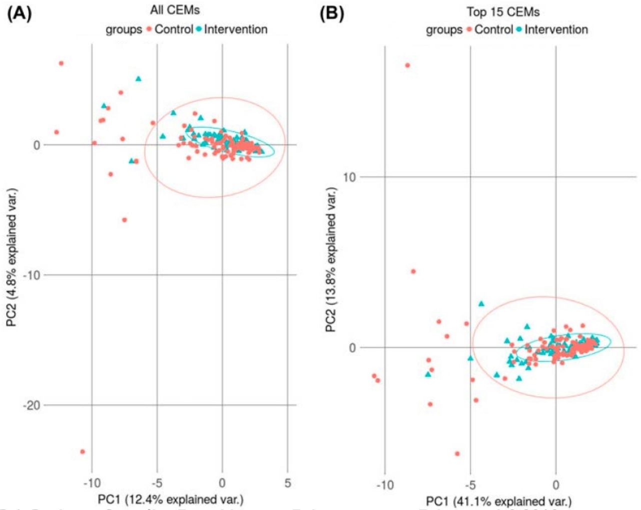
comparisons are difficult

- Denny categorized CEMs, Becker did not
- comparing Denny's three CEMs with Becker - which is not a perfect comparison:
  - agree on *; not found* - pretty straightforward CEM
  - disagree on other two
- remember - we can only compare 3 CEMs/categories for these two studies as Denny *et al.* (2011) only studied 3

See goo.gl/sJCPRF for a detailed discussion on comparing 'similar' studies.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker and Mooney (2016)



Principal component analysis showing increased variability of control group. (A) All CEMs. (B) Top 15 CEMs (representing 85% of all errors).

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Pettit *et al.* (2017)

setup

- Control/Treatment
- Editor/Language: Athene/C++
- Students: unknown (but a lot)
- Errors: unknown (but a lot) - 36,000 submissions
- Time: 8 semesters
- Particulars
  - Verbosely enhanced ECEMs
  - ECEMs shown side-by-side with CEMs (as Becker)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Pettit *et al.* (2017)

results

- **no** effect of ECEMs on the following metrics:
  - likelihood of successive compilation **errors**
  - occurrence of compiler **errors** within semesters
  - **student progress** towards a successful completion of a programming assignment
- student questionnaire results were positive

Prather *et al.* (2017)

approach

- applied human-factors approach to investigating effects of ECEMs (including think-alouds)
- reviewed Becker, Denny, Pettit and offered 5 possible reasons for inconsistent conclusions regarding ECEMs:
  1. students don't read them
  2. researchers measuring wrong thing
  3. effects are hard to measure
  4. messages are not properly designed
  5. messages are properly designed, but students don't understand ECEMs due to high cognitive load
- conducted mixed-methods experiments to address 1-5 above

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Prather *et al.* (2017)

results

- students do read ECEMs
- in low cognitive load settings, ECEMs seem to be understood
- in high cognitive load settings:
  - quantitative evidence was inconclusive
  - qualitative evidence indicated that ECEMs were helpful

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Harker (2017)

approach

- MSc (by research)
- used slightly updated version of Decaf (Becker *et al.*)
- partial replication of Becker (2015)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Harker (2017)
setup

- Control/Treatment
- Editor/Language: Decaf/Java
- Students: approx 50
- Errors: several thousand
- Time: 1 semester
- Particulars
  - used rubric of Marceau *et al.* (2011a) (slightly extended)
  - ECEMs shown side-by-side with CEMs (as Becker)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Harker (2017)

results

- **positive** effect of ECEMs on the following metrics
  - number of failed **compilations** during assignments
  - number of failed **compilations** during laboratory quizzes
  - **time** spent fixing errors
- **no** effect on **academic performance**
- student questionnaire responses also promising

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2018a)

approach

- control (CEM) and treatment (ECEM) groups both took a test consisting of a program containing 24 syntax errors
- objective was to rectify all errors
- test was logically made up of 5 separate tasks with a specified, known output (when errors are resolved)
  - forced some consistency in error resolution
  - for instance, this eliminated 'fixing' an error by just deleting the line of code

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2018a)

setup

- Control/Treatment
- Editor/Language: Decaf/Java
- Students: approx 100
- Errors: approx 2000
- Time: 2 hours
- Particulars
    - ECEMs shown side-by-side with CEMs

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Becker *et al.* (2018a)

results

1. **no** effect on the number of compiling **submissions**
2. **no** effect on the number of completed **tasks**
3. **no** effect on student **scores**
4. **positive** effect on number of **errors** rectified

This study agreed with many previous studies, both where effects were found and where effects were not found:

1. $\rightarrow$ Denny *et al.* (2014)
2. $\rightarrow$ Harrington (1984)
3. $\rightarrow$ Harker (2017); Pettit *et al.* (2017)
4. $\rightarrow$ Becker (2015); Becker *et al.* (2016); Becker (2016); Becker *et al.* (2018a); Harker (2017)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

What are we measuring?

2010 - present: 6+ groups, 12+ papers and 2 theses, measuring at least 10 metrics:

- error frequency, overall number of errors, errors per student, repeated errors
- attempts required to resolve errors
- non-compiling submissions
- student progress, test scores, academic performance
- time to resolve errors

Both positive and negative evidence, some corroborated, some not

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Weak signals?

It is possible that ECEMs have effects but the signal is weak.

- metrics measuring errors (number of, frequency of, time to fix, etc.) are relatively 'close to the data'
  - we are seeing evidence of effects at this distance
  - all studies with large numbers of errors seem to agree
- metrics measuring academic performance, submissions, tests, etc. are arguably 'farther from the data'.
  - we are not seeing evidence of effects at these distances
  - or, we haven't seen corroboration $\rightarrow$ more studies needed!
  - or, *continued...*

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

or...

...we shouldn't be looking for weak signals far from the source

- Marceau *et al.* (2011a)
  - "these studies suffer from having too many confounding factors underneath their dependent variable"[6]
  - 'the performance of the error messages should be measured, not the performance of the students'

[6] speaking of pre-2011 studies

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

What metrics do we need to measure

Is a reduction in frequency of errors (e.g. Becker *et al.*, Harker) good enough, regardless of effects on performance, etc.?

- What are the effects of a reduction in the frequency of errors?
    - Possibly good psychological effects - efficacy, etc.
    - **Frustration** with CEMs is cropping up in a lot of places (Becker, 2015; Becker *et al.*, 2018b; Harker, 2017; Pettit *et al.*, 2017; Prather *et al.*, 2017)
    - Should we be aiming to measure frustration (difficult!) instead of improved academic performance? Would reducing frustration improve performance?
- *continued...*

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

What metrics do we need to measure

- What about studies that at report a better experience for students?
- They seem to indicate that ECEMs are perceived as helpful (Becker, 2015; Pettit *et al.*, 2017; Prather *et al.*, 2017; Harker, 2017).
- This could potentially improve self-efficacy, reduce frustration, and possibly reduce early drop-outs.

Should we abandon empirical quantitative ’technical’ metrics and focus on empirical qualitative ’human/learning/perception’ metrics?

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Show me the (enhanced) error messages!

Example CEM/ECEM pair (text only, no structure, line numbers, etc.):

- CEM: *undefined variable*
- ECEM: *on the specified line there is a mis-spelled variable name or a missing variable declaration. Check spelling and that you are not using a variable that is defined previously.*

Harker (2017) provides the most recent and most comprehensive list of Java CEMs/ECEMs.

- not online yet, but I can give you a copy
- I also have several (Java) lists that I can share (they tend to 'evolve' slowly for several reasons)

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Health Warning

Generalizing these / crossing language borders, etc. is fraught with many 'small' but difficult issues, as Brett and Stefik found out this week.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Progress straight out of the box

Compilers and languages

- Eclipse has its own compiler for Java with CEMs different to the JDK
- Why are these CEMs different?
  - Are these ECEMs?
  - Ben-Ari (2007) states that these are 'superior' to javac.
- Some new(er) languages are striving to provide better CEMs:
  - Clang[7]
  - Elm[8]
  - Racket (Marceau *et al.*, 2011a)
  - Quorum[9]

[7] clang.llvm.org/diagnostics.html
[8] elm-lang.org/blog/compiler-errors-for-humans
[9] quorumlanguage.com/evidence.html

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Frameworks, guidelines, principles, no-nos, rubrics, etc.

- Traver (2010) introduced principles for compiler error design
- Coull and Duncan (2011) introduced a conceptual framework for software tool development that includes CEM considerations
- What to do and what not to do (at least, historically): (Wexelblat, 1976; McIver and Conway, 1996)
    - Are we still doing the no-nos and are we actually doing the should-dos?
- Marceau *et al.* (2011a) introduced a language-agnostic rubric for evaluating the effectiveness of enhanced compiler error messages. Their premise was that the rubric should assess the performance of the error messages, not the students. Also see Marceau *et al.* (2011b).

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Frameworks, guidelines, principles, no-nos, rubrics, etc.

- Wrenn and Krishnamurthi (2017) presented a lightweight process to guide error report authoring from the perspective that error reports are really classifiers of program information.
  - they should be subjected to the same measures as other classifiers (e.g., precision and recall).
  - the authors formalized this perspective as a process for assessing error reports, applied the process to an actual programming language, and presented a preliminary study on the utility of the resulting error reports.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Moving forward

- Poor compiler error messages are a problem.
  - Althouh I have focused on novices, they can also be a problem for more experienced programmers.
- Do we need to 'go back to the basics'?
- Perhaps we should be looking at the readability of CEMs/ECEMs.
- Surely better compiler error messages are a good thing regardless of effect (as long as effect is not bad)?
  - Should we be aiming to reduce frustration?
  - Should we be focusing more on the user experience?
- What about incremental 'red squiggly lines' compilers?
  - How do these affect novices?
  - BlueJ has moved to this model...

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Summary

- many (most?) compiler error messages are poor, and do not have much (UX, psychological, lingusitic, etc.) thought behind them
- they cause trouble, technically, and psychologically
    - they do not need to
- what are the effects of 'good' error messages?
    - provided they do no harm, are they better than poor messages, even without demonstrated improvements in metrics?
- how can we design them to be more effective, and measure that effectiveness?
- work has begun, but more work is needed

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

References

Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E., and Parnin, C. (2017). Do developers read compiler error messages? ICSE '17, pages 575–585.

Becker, B. A. (2015). An exploration of the effects of enhanced compiler error messages for computer programming novices. Master's thesis, Dublin Institute of Technology.

Becker, B. A. (2016). An effective approach to enhancing compiler error messages. In SIGCSE '16, pages 126–131. ACM.

Becker, B. A. and Mooney, C. (2016). Categorizing compiler error messages with principal component analysis. In 12th China-Europe International Symposium on Software Engineering Education (CEISEE 2016), Shenyang, China, 28-29 May 2016.

Becker, B. A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K., and Mooney, C. (2016). Effective compiler error message enhancement for novice programming students. Computer Science Education, **26**(2-3), 148–175.

Becker, B. A., Goslin, K., and Glanville, G. (2018a). The effects of enhanced compiler error messages on a syntax error debugging test. In SIGCSE '18. ACM.

Becker, B. A., Murray, C., Tianyi, T., Song, C., McCartney, R., and Sanders, K. (2018b). Fix the first, ignore the rest: Dealing with multiple compiler error messages. In SIGCSE '18. ACM.

Ben-Ari, M. (2007). Compile and Runtime Errors in Java. `http://www.weizmann.ac.il/sci-tea/benari/software-and-learning-materials/errors.pdf`. Accessed 12 Jan 2017.

Brown, N. C. C., Kölling, M., McCall, D., and Utting, I. (2014). Blackbox: A large scale repository of novice programmers' activity. In SIGCSE '14, pages 223–228. ACM.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

References

Brown, P. J. (1983). Error messages: The neglected area of the man/machine interface. Communications of the ACM, **26**(4), 246–249.

Coull, N. J. and Duncan, I. M. (2011). Emergent requirements for supporting introductory programming. Innovation in Teaching and Learning in Information and Computer Sciences, **10**(1), 78–85.

Denny, P., Luxton-Reilly, A., Tempero, E., and Hendrickx, J. (2011). Codewrite: Supporting student-driven practice of Java. In SIGCSE '11, pages 471–476. ACM.

Denny, P., Luxton-Reilly, A., and Tempero, E. (2012). All syntax errors are not equal. In ITiCSE '12, pages 75–80. ACM.

Denny, P., Luxton-Reilly, A., and Carpenter, D. (2014). Enhancing syntax error messages appears ineffectual. In ITiCSE '14, pages 273–278. ACM.

Du Boulay, B. and Matthew, I. (1984). Fatal error in pass zero: How not to confuse novices. In Readings on Cognitive Ergonomics—Mind and Computers, pages 132–141. Springer.

Dy, T. and Rodrigo, M. M. (2010). A detector for non-literal java errors. In Koli Calling '10, pages 118–122. ACM.

Flowers, T., Carver, C., and Jackson, J. (2004). Empowering students and building confidence in novice programmers through Gauntlet. In FIE '04, pages T3H–10. IEEE.

Harker, D. (2017). Examining the effects of enhanced compilers on student productivity. Master's thesis, University of Northern British Columbia.

Harrington, J. L. (1984). The Effect of Error Messages on Learning Computer Programming by Individuals Without Prior Programming Experience. Ph.D. thesis, Drexel University.

Horning, J. J. (1974). What the compiler should tell the user, pages 525–548. Springer Berlin Heidelberg.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

References

Jackson, J., Cobb, M., and Carver, C. (2005). Identifying top java errors for novice programmers. In Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference, pages T4C–T4C. IEEE.

Jadud, M. C. (2006). An exploration of novice compilation behaviour in BlueJ. Ph.D. thesis, University of Kent.

Ko, A. (March 25, 2014 (accessed August 6, 2017)). Programming languages are the least usable, but most powerful human-computer interfaces ever invented. http://blogs.uw.edu/ajko/2014/03/25/programming-languages-are-the-least-usable-but-most-powerful-human-computer-interfaces-ever-invented/.

Kummerfeld, S. K. and Kay, J. (2003). The neglected battle fields of syntax errors. In Proceedings of the fifth Australasian conference on Computing education-Volume 20, pages 105–111. Australian Computer Society, Inc.

Lane, H. C. and VanLehn, K. (2005). Intention-based scoring: an approach to measuring success at solving the composition problem. In ACM SIGCSE Bulletin, volume 37, pages 373–377. ACM.

Lee, M. J. and Ko, A. J. (2011). Personifying programming tool feedback improves novice programmers' learning. In Proceedings of the seventh international workshop on Computing education research, pages 109–116. ACM.

Litecky, C. R. and Davis, G. B. (1976). A study of errors, error-proneness, and error diagnosis in cobol. Communications of the ACM, **19**(1), 33–38.

Marceau, G., Fisler, K., and Krishnamurthi, S. (2011a). Measuring the effectiveness of error messages designed for novice programmers. In SIGCSE '11, pages 499–504. ACM.

Marceau, G., Fisler, K., and Krishnamurthi, S. (2011b). Mind your language: On novices' interactions with error messages. In SIGPLAN Onward! '11, pages 3–18. ACM.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

References

McCall, D. and Kölling, M. (2014). Meaningful categorisation of novice programmer errors. In 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, pages 1–8. IEEE.

McIver, L. and Conway, D. (1996). Seven deadly sins of introductory programming language design. In Software Engineering: Education and Practice, 1996. Proceedings. International Conference, pages 309–316. IEEE.

Moulton, P. and Muller, M. (1967). Ditran—a compiler emphasizing diagnostics. Communications of the ACM, **10**(1), 45–52.

Nienaltowski, M.-H., Pedroni, M., and Meyer, B. (2008). Compiler error messages: What can help novices? In ACM SIGCSE Bulletin, volume 40, pages 168–172. ACM.

Pettit, R. S., Homer, J., and Gee, R. (2017). Do enhanced compiler error messages help students?: Results inconclusive. In SIGCSE '17, pages 465–470. ACM.

Prather, J., Pettit, R., McMurry, K. H., Peters, A., Homer, J., Simone, N., and Cohen, M. (2017). On novices' interaction with compiler error messages: A human factors approach. In ICER '17, pages 74–82, New York, NY, USA. ACM.

Seo, H., Sadowski, C., Elbaum, S., Aftandilian, E., and Bowdidge, R. (2014). Programmers' build errors: a case study (at google). In Proceedings of the 36th International Conference on Software Engineering, pages 724–734. ACM.

Stefik, A. and Siebert, S. (2013). An empirical investigation into programming language syntax. ACM Transactions on Computing Education (TOCE), **13**(4), 19.

Tabanao, E. S., Rodrigo, M. M. T., and Jadud, M. C. (2011). Predicting at-risk novice java programmers through the analysis of online protocols. In Proceedings of the seventh international workshop on Computing education research, pages 85–92. ACM.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

References

Thompson, S. M. (2004). An exploratory study of novice programming experiences and errors. Master's thesis, University of Victoria.

Traver, V. J. (2010). On compiler error messages: What they say and what they mean. Advances in Human-Computer Interaction, **2010**, 3:1–3:26.

Wexelblat, R. L. (1976). Maxims for malfeasant designers, or how to design languages to make programming as difficult as possible. In ICSE '76, ICSE '76, pages 331–336, Los Alamitos, CA, USA. IEEE Computer Society Press.

Wrenn, J. and Krishnamurthi, S. (2017). Error messages are classifiers. In Proceedings of 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software Onward17. ACM.

University College Dublin
An Coláiste Ollscoile Baile átha Cliath

Academic archaeology

Harrington (1984), retrieved from Drexel University library on microfilm, delivered in a paper envelope with a stamp...

- enhanced Applesoft BASIC errors
- 39 subjects randomly assigned to three 'error levels'
  1. standard Applesoft BASIC CEMs
  2. ECEMs - additional parsing to identify and describe CEMs more specifically
  3. ECEMs - even more specific
- two 'task' sessions - one debugging, one programming
- sessions analyzed for number of errors, average attempts to correct, success on task
- result: 'merely increasing the level of specificity of ECEMs does not necessarily decrease debugging effort or increase ultimate success'