ON IMPACT IN SOFTWARE ENGINEERING RESEARCH

ANDREAS ZELLER, CISPA / SAARLAND UNIVERSITY

ICSE NEW FACULTY SYMPOSIUM
GÖTEBORG, MAY 29, 2018
ANDREAS ZELLER: KEY FACTS

• PhD in 1997 on Configuration Management with Feature Logic

• Since 2001 professor at Saarland Informatics Campus (Saarland University / CISPA)

• Four 10-year impact awards 2009–2017 (for papers 1999–2007)

• ACM Fellow in 2010

• ERC Advanced Grant in 2011

• SIGSOFT Outstanding Research Award on Friday
WHAT IS IMPACT?
WHAT IS IMPACT?

- How do your actions change the world?
- Often measured in citations, publications, funding, people, ...
- All these are indicators of impact, but not goals in themselves
- We want to make the world a better place
- Gives meaning and purpose to our (professional) life
WHAT MAKES IMPACTFUL RESEARCH?

• *Intellectual challenge* – was it hard, or could anyone have done this?

• *Elegance* – is your research specific to a context, or can it be reused again and again?

• *Usefulness* – can someone make money with it?

• Innovation is the *delta* in any of these metrics
VARYING PERSPECTIVES

• *Programming Languages* folks miss the intellectual challenge

• *Formal Methods* folks miss elegance and challenge

• *Industry* folks miss usefulness and applicability
WHAT MAKES IMPACTFUL RESEARCH?

• How did your work make the world a better place?
MY PATH TO IMPACT
MY PATH TO IMPACT

• Life can only be understood backwards; but it must be lived forwards (Søren Kierkegaard)
CONFIGURATION MANAGEMENT WITH FEATURE LOGIC (1991-1997)

- Topic defined by my PhD advisor Gregor Snelting
- Idea: Formally describe variants and revisions with feature logic
- “A unified model for configuration management”

Figure 8: Delta features and other features
FEATURE LOGIC: LESSONS LEARNED

• You can get plenty of papers accepted
  • even if you miss the problem
  • even if you do not evaluate
• “Modeling for the sake of modeling”
• Enabled much of my later work, though
DDD (1994-1999)

- During PhD, programmed a lot
- Debugging was hard!
- Built the DDD debugger GUI with my student Dorothea Lütkehaus
- Welcome change from formal work
DDD (1994-1999)

- DDD was among the first development tools with a “professional” GUI
- Downloaded by the tens of thousands
- Adopted as a GNU project: Street credibility with developers
- Impact through usefulness
DDD: LESSONS LEARNED

• Work on a real problem
  - "real" as in "real world", not "real papers"

• Assume as little as possible
  - make things fit into real processes

• Keep things simple
  - complexity impresses, but prevents impact

• After PhD, looking for new topic
• Delta Debugging brought together debugging and version control
• Isolate failure causes through repeated experiments

- Delta debugging was a bomb
- Easy to teach + understand
- 7 lines of algorithm (and 25 lines of Python)
- Spent two years on these

\[
\begin{align*}
\dd(c_\bigvee, c_\bigwedge) &= \dd' (c_\bigvee, c_\bigwedge, 2) \\
\dd' (c'_\bigvee, c'_\bigwedge, n) &= \\
&\begin{cases} 
(c'_\bigvee, c'_\bigwedge) & \text{if } |\Delta| = 1 \\
\dd' (c'_\bigvee \setminus \Delta_i, c'_\bigwedge, 2) & \text{if } \exists i \in \{1..n\} \cdot \text{test}(c'_\bigwedge \setminus \Delta_i) = \bigvee \\
\dd' (c'_\bigvee, c'_\bigwedge \cup \Delta_i, 2) & \text{if } \exists i \in \{1..n\} \cdot \text{test}(c'_\bigwedge \cup \Delta_i) = \bigtimes \\
\dd' (c'_\bigvee \cup \Delta_i, c'_\bigwedge, \max(n - 1, 2)) & \text{else if } \exists i \in \{1..n\} \cdot \text{test}(c'_\bigwedge \cup \Delta_i) = \bigvee \\
\dd' (c'_\bigvee, c'_\bigwedge \setminus \Delta_i, \max(n - 1, 2)) & \text{else if } \exists i \in \{1..n\} \cdot \text{test}(c'_\bigwedge \setminus \Delta_i) = \bigtimes \\
\dd' (c'_\bigvee, c'_\bigwedge, \min(2n, |\Delta|)) & \text{else if } n < |\Delta| \text{ (“increase granularity”) } \\
(c'_\bigvee, c'_\bigwedge) & \text{otherwise}
\end{cases}
\end{align*}
\]
DELTA DEBUGGING: LESSONS LEARNED

- Work on a real problem
  - Why debug? We build correct software
- Assume as little as possible
  - Version control? tests? Never heard of it
- Keep things simple
  - 25 lines of Python is probably excessive
- Have a sound model
  - DD was my version model reborn
MINING SOFTWARE ARCHIVES (2003-2010)

- In the early 2000s, open-source version repositories became available
- Stephan Diehl saw an opportunity for visualization and approached me
- Quickly expanded into data mining
- Tom Zimmermann: our MSc student
- Work of a research team
MINING SOFTWARE ARCHIVES (2003-2010)

• Our 2004 paper was the first ICSE paper on mining software archives

• Handful of competing groups; instant hit

• MSR now a conference on its own

• Paper has 1200+ citations so far

• Impact at Microsoft, Google, SAP...
MINING SOFTWARE ARCHIVES (2003-2010)

- We are now after the gold rush
- Data still exciting (if you have some)
- Few new insights on old data
- Get out of a field when too crowded
MINING SOFTWARE REPOSITORIES: LESSONS LEARNED

• Work on a real problem
  - *Empirical research is core field of SE*

• Assume as little as possible
  - *simple parsers for multiple languages*

• Keep things simple
  - *essence of 2004 paper is one line of SQL*

• Have a sound model
  - *retrieval, precision, recall, etc, etc*

• Keep on learning
  - *statistics, data mining, machine learning*
• How do we know an app does what it should do?
• CHABADA checks for mismatches between description and behavior
• Novel usage of NLP; novel app store mining
• The ICSE paper of 2014 is among most cited
• CHABADA techniques now adopted by Google and Microsoft
• Most of your mobile apps have gone through such an analysis :-)
MINING APPS: LESSONS LEARNED

- Work on a real problem
- Assume as little as possible
- Keep things simple
- Have a sound model
- Keep on learning
- Keep on moving

- Yes, there is malware
- Descriptions and APIs
- Standard NLP techniques
- Standard NLP methods
- NLP, machine learning, recommendation...
- Security starts with SE
MORE THINGS I DID

- Automatic repair
  - Wesley Weimer beat us to it
- Automatic parallelization
  - Struggled with complexity
- Automatic website testing
  - Built a company for that
- Structured fuzzing
  - Langfuzz found 2000+ browser bugs
- Automatic sandboxing
  - Lots of potential in here
THINGS I STAYED AWAY FROM

• Symbolic techniques
• Formal methods
• Modeling
• Architecture

• Work on a real problem
• Assume as little as possible
• Keep things simple
• Have a sound model
• Keep on learning
• Keep on moving
YOUR WAYS TO HAVE IMPACT
IMPACT AS A RESEARCHER

• Society funds research to take risks that no one else does

• Research is risky by construction – you should expect to fail, and fail again

• Tenure is meant to allow you to take arbitrarily grand challenges - so work on the grand stuff

• If you lack resources, try smarter and harder
IMPACT AS A TEACHER

• Teaching can be a great way to multiply your message

• Not only focus on teaching the standards, but also your research

• Teaching your research helps to propagate it and make it accessible

• Engage students on topics dear to you
IMPACT WITH INDUSTRY

• Do work with industry to find problems and frame your work

• Do not work with industry to solve (their) concrete problems

• Your role as researcher is more than a cheap consulting tool

• Many “research” funding schemes are there to subsidize industry
IMPACT THROUGH TOOLS

• Getting your technique out as a tool is a great way to have impact!

• Also allows to check what actual users need (and if they exist)

• A tool can have far more impact than a paper

• Funding agencies and hiring committees begin to realize this
IMPACT AS FOUNDER

• Creating a company out of your research can be great fun!

• Push your research and ideas into practice

• Again, shows you what the market wants (and what not)

• Plenty of support available (money, consultancy)
IMPACT AS MENTOR

• Working with advanced students can be the most satisfying part of your job

• The variety of SE research needs universal problem solving skills

• Find such skills besides good grades
A GREAT ENVIRONMENT

• My institution (Saarland University) hired me although I was the candidate with the fewest publications

• But they liked the papers, so they hired me

• No pressure or incentives on papers, citations, funding, etc.

• One single expectation: long-term impact
SURVIVOR BIAS

- Researchers with great impact are the selected few who survived academic selection
- What worked for me will not work for most
- Most of us have to struggle with plenty of bad, misguided, short-term career incentives
- Follow incentives until tenured, then set better ones
- Get lucky!
ON IMPACT IN SOFTWARE ENGINEERING RESEARCH

ANDREAS ZELLER, CISPA / SAARLAND UNIVERSITY
LESSONS LEARNED:
ON IMPACT IN SE RESEARCH

• Work on a real problem
  - possibly bursting your bubble
• Assume as little as possible
  - immediate impact on adoption
• Keep things simple
  - complexity inhibits impact
• Have a sound model
  - tools may fade away, concepts persist
• Keep on learning
  - learn new stuff and leverage it
• Keep on moving
  - do not stay in your cozy SE corner