Evaluation

Three Categories of User Study RQs

Need Finding Study

What are interesting **problems** to solve?

Formative Study

For a given **problem**, what are promising **solutions**?

Evaluative Study

For a given **problem**, now that we've implemented a **solution**, did it work?

Three Categories of User Study RQs

Need Finding Study

What are interesting **problems** to solve?

A long while back now

Formative Study

For a given **problem**, what are promising **solutions**?

Evaluative Study

For a given **problem**, now that we've implemented a **solution**, did it work?

Shape of a Need Finding RQ

What kinds of problems does <description of audience> face during <description of tasks>?

For this class, usually…

What kinds of problems does <description of audience> face during <description of *programming* tasks>?

Three Categories of User Study RQs

Need Finding Study

What are interesting **problems** to solve?

Formative Study

For a given **problem**, what are promising **solutions**?

Still a short while back

Evaluative Study

For a given **problem**, now that we've implemented a **solution**, did it work?

Examples of Formative RQ

Now we’re trying to build something, some kind of intervention. So what do we need to know about users in order to build interventions that will actually serve them?

…

Are <description of audience> more <successful/efficient/bug-free/comfortable> doing interaction X or interaction Y?

When <description of audience> describe <description of task> to a human assistant, how do they explain the task spec?

When we ask <description of audience> to do <description of task> with <tool>, what goes wrong?

Three Categories of User Study RQs

Need Finding Study

What are interesting **problems** to solve?

Formative Study

For a given **problem**, what are promising **solutions**?

Evaluative Study

For a given **problem**, now that we've implemented a **solution**, did it work?

This week

Shape of an Evaluative RQ

Does <tool> <increase/decrease> <measure of success> for <description of audience> doing <description of tasks>?

Examples:

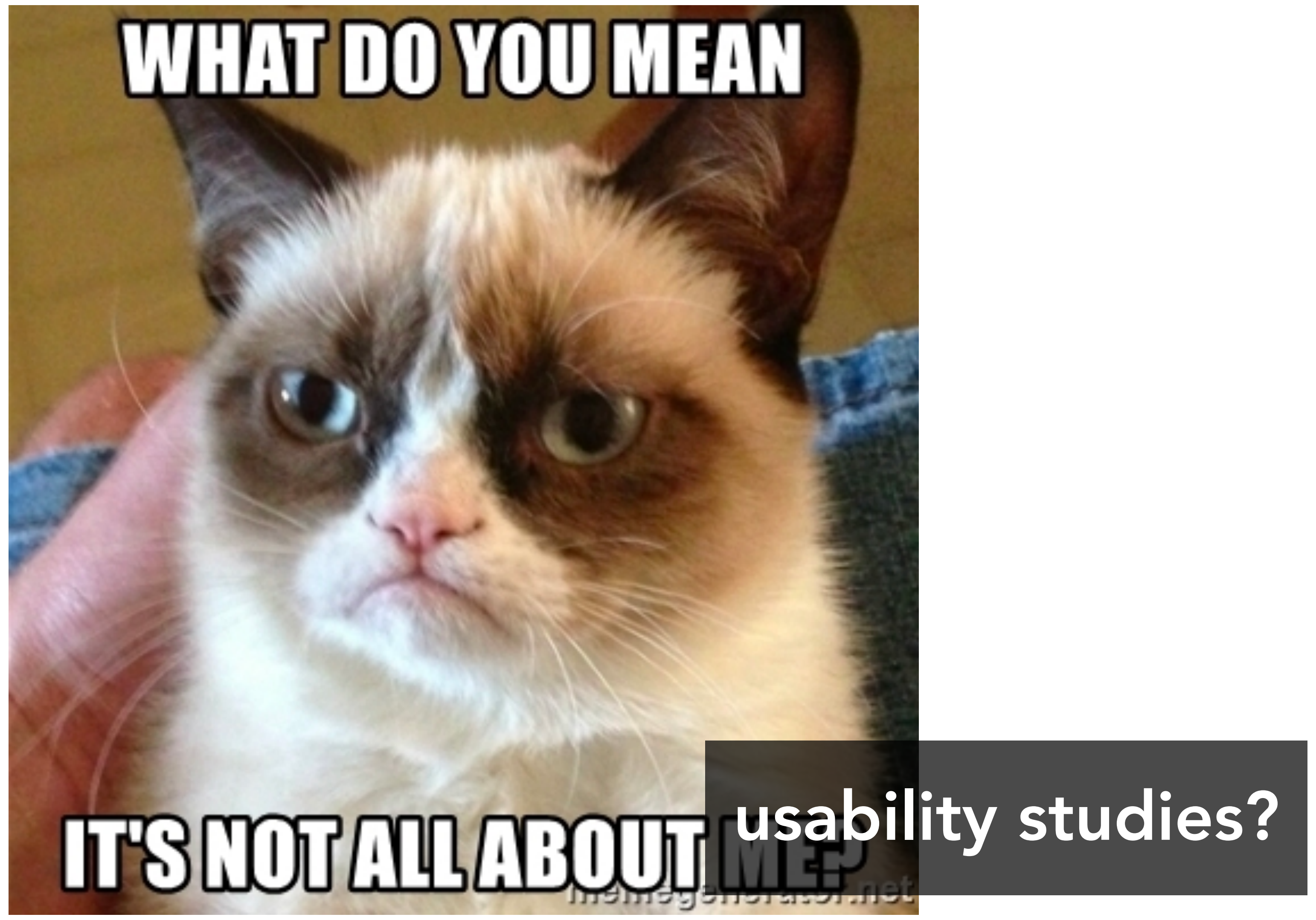
Does our domain-specific language decrease time-to-correct-program for first-year CS students doing data extraction from scanned forms?

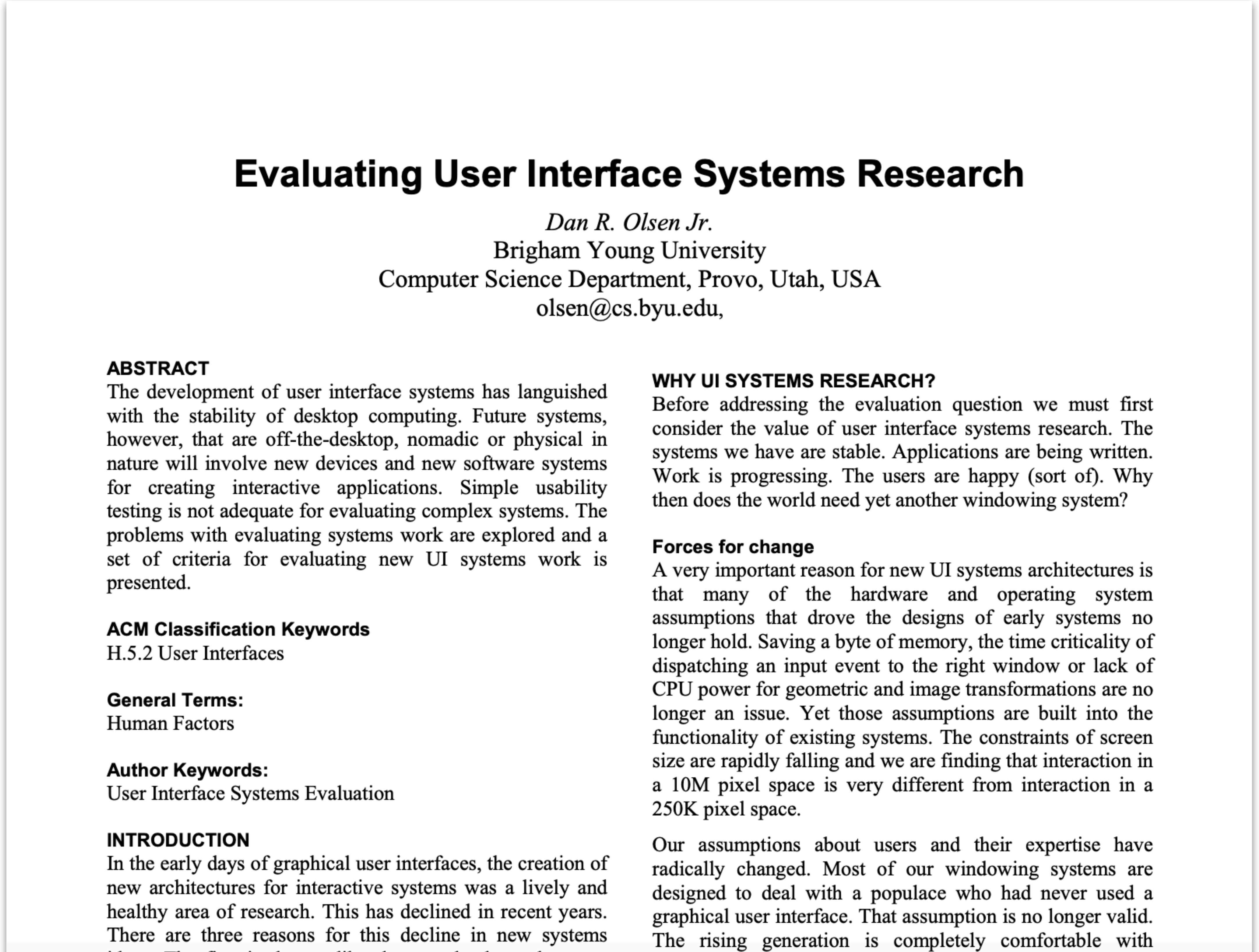
Does our program synthesis tool increase chatbot friendliness ratings for psychologists doing chatbot implementation?

Does our custom programming environment decrease debugging time for biostatisticians doing public health analyses on large datasets?

Plan for today

A structured conversation about the relationship between today’s reading and our role as PL+HCI researchers





Evaluating User Interface Systems Research

Dan R. Olsen Jr.
Brigham Young University
Computer Science Department, Provo, Utah, USA
olsen@cs.byu.edu,

ABSTRACT
The development of user interface systems has languished with the stability of desktop computing. Future systems, however, that are off-the-desktop, nomadic or physical in nature will involve new devices and new software systems for creating interactive applications. Simple usability testing is not adequate for evaluating complex systems. The problems with evaluating systems work are explored and a set of criteria for evaluating new UI systems work is presented.

ACM Classification Keywords
H.5.2 User Interfaces

General Terms:
Human Factors

Author Keywords:
User Interface Systems Evaluation

INTRODUCTION
In the early days of graphical user interfaces, the creation of new architectures for interactive systems was a lively and healthy area of research. This has declined in recent years. There are three reasons for this decline in new systems

WHY UI SYSTEMS RESEARCH?
Before addressing the evaluation question we must first consider the value of user interface systems research. The systems we have are stable. Applications are being written. Work is progressing. The users are happy (sort of). Why then does the world need yet another windowing system?

Forces for change
A very important reason for new UI systems architectures is that many of the hardware and operating system assumptions that drove the designs of early systems no longer hold. Saving a byte of memory, the time criticality of dispatching an input event to the right window or lack of CPU power for geometric and image transformations are no longer an issue. Yet those assumptions are built into the functionality of existing systems. The constraints of screen size are rapidly falling and we are finding that interaction in a 10M pixel space is very different from interaction in a 250K pixel space.

Our assumptions about users and their expertise have radically changed. Most of our windowing systems are designed to deal with a populace who had never used a graphical user interface. That assumption is no longer valid. The rising generation is completely comfortable with

This paper played a big role in the HCI community in broadening the classes of evaluations considered acceptable, including no-evaluation papers.

What's this to do with us?

- A lot of parallels to evaluating PLs. (In your head, replace "UI system" or "UI toolkit" with "PL" and see how many observations still hold.)
- Framework for how to think about meaningfully evaluating complex design contributions

Thank Amy Ko for these insights, and check out her work for more of the same!

Value added by UI systems architecture (…and PLs!)

- Reduce development viscosity
- Least resistance to good solutions
- Lower skill barriers
- Power in common infrastructure
- Enabling scale

Evaluation Errors

- The usability trap
- The fatal flaw fallacy
- Legacy code

Usability Trap

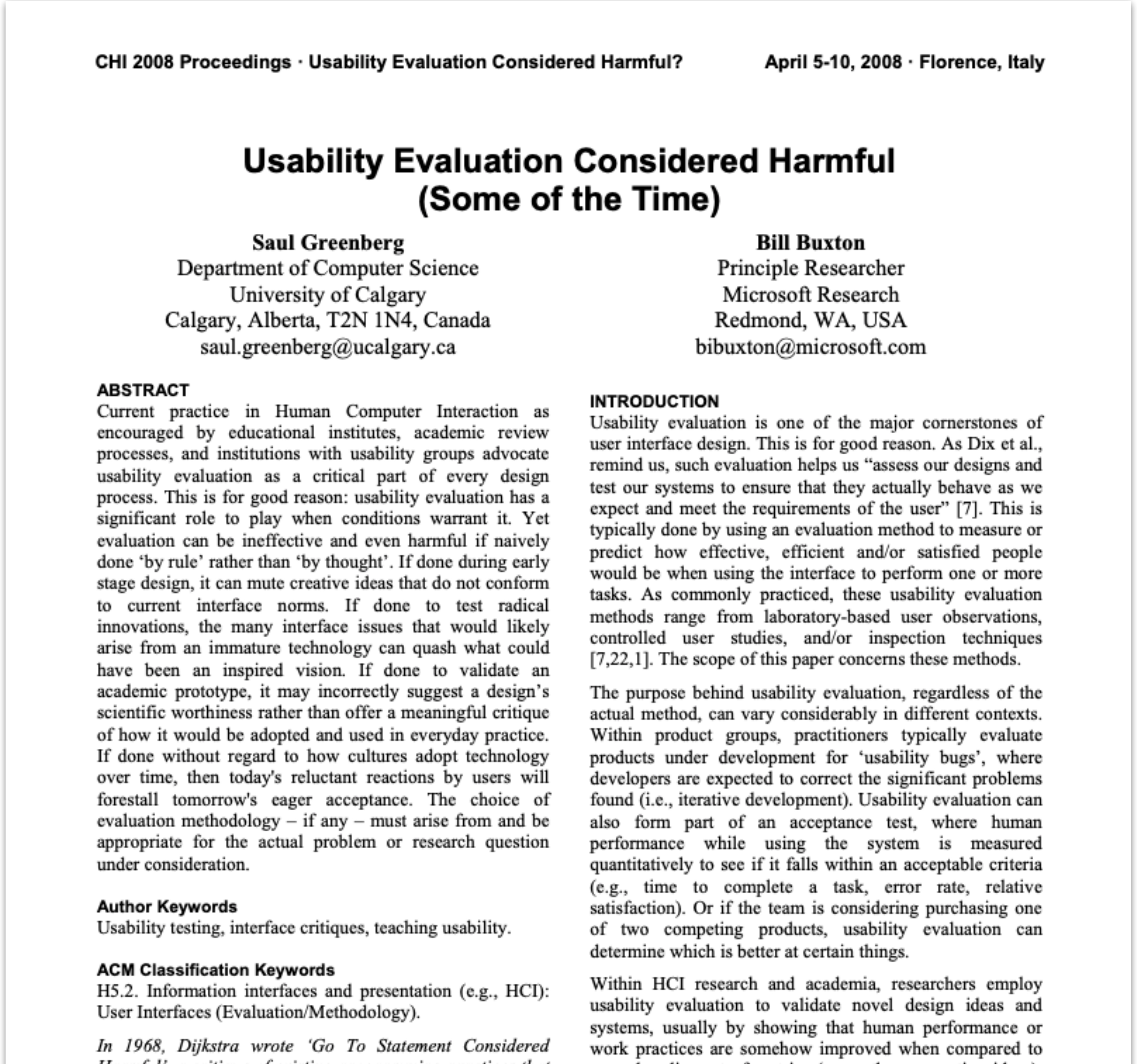
Common measures

- Time to complete standard task
- Time to reach proficiency
- Number of errors

Sound familiar?

Another take on the usability trap, well worth a read

- Usability eval as weak science
  - Do we end up picking problems and solutions that are amenable to these evals rather than picking research question, then choosing eval that fits?
  - We often do this rather than testing risky hypothesis.
- Using usability eval too early
  - Quashing cool ideas by testing for usability before they're usable, even if they have promise
  - Consider too few ideas; many parallel ideas standard in other design and engineering fields
- Innovation, Cultural Adoption
  - **Usable vs. useful**
  - Discovery: find facts about the world
  - Innovation, invention: create new and useful things
  - Many very useful inventions (e.g., cars) started out pretty unusable
  - Even our best inventors often don't anticipate how culture will use the inventions



CHI 2008 Proceedings · Usability Evaluation Considered Harmful? **April 5-10, 2008 · Florence, Italy**

Usability Evaluation Considered Harmful (Some of the Time)

Saul Greenberg
Department of Computer Science
University of Calgary
Calgary, Alberta, T2N 1N4, Canada
saul.greenberg@ucalgary.ca

Bill Buxton
Principle Researcher
Microsoft Research
Redmond, WA, USA
bibuxton@microsoft.com

ABSTRACT

Current practice in Human Computer Interaction as encouraged by educational institutes, academic review processes, and institutions with usability groups advocate usability evaluation as a critical part of every design process. This is for good reason: usability evaluation has a significant role to play when conditions warrant it. Yet evaluation can be ineffective and even harmful if naively done 'by rule' rather than 'by thought'. If done during early stage design, it can mute creative ideas that do not conform to current interface norms. If done to test radical innovations, the many interface issues that would likely arise from an immature technology can quash what could have been an inspired vision. If done to validate an academic prototype, it may incorrectly suggest a design's scientific worthiness rather than offer a meaningful critique of how it would be adopted and used in everyday practice. If done without regard to how cultures adopt technology over time, then today's reluctant reactions by users will forestall tomorrow's eager acceptance. The choice of evaluation methodology – if any – must arise from and be appropriate for the actual problem or research question under consideration.

Author Keywords
Usability testing, interface critiques, teaching usability.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces (Evaluation/Methodology).

In 1968, Dijkstra wrote 'Go To Statement Considered

INTRODUCTION

Usability evaluation is one of the major cornerstones of user interface design. This is for good reason. As Dix et al., remind us, such evaluation helps us "assess our designs and test our systems to ensure that they actually behave as we expect and meet the requirements of the user" [7]. This is typically done by using an evaluation method to measure or predict how effective, efficient and/or satisfied people would be when using the interface to perform one or more tasks. As commonly practiced, these usability evaluation methods range from laboratory-based user observations, controlled user studies, and/or inspection techniques [7,22,1]. The scope of this paper concerns these methods.

The purpose behind usability evaluation, regardless of the actual method, can vary considerably in different contexts. Within product groups, practitioners typically evaluate products under development for 'usability bugs', where developers are expected to correct the significant problems found (i.e., iterative development). Usability evaluation can also form part of an acceptance test, where human performance while using the system is measured quantitatively to see if it falls within an acceptable criteria (e.g., time to complete a task, error rate, relative satisfaction). Or if the team is considering purchasing one of two competing products, usability evaluation can determine which is better at certain things.

Within HCI research and academia, researchers employ usability evaluation to validate novel design ideas and systems, usually by showing that human performance or work practices are somehow improved when compared to

Usability Trap

Let’s chat!

Common assumptions

- Walk up and use, minimal training
  - Using doesn’t require expertise, or if it requires specific expertise many people already have that expertise
- Standardized task assumption
  - If we’re going to compare across two systems, there has to be an existing system that can already accomplish the task
- Scale of the problem
  - Task usually needs to be completable in 1-2 hours

Let’s chat!

The fatal flaw fallacy

Say every time someone proposes a new PL or new abstraction, we try to find a program that can’t be expressed with it. Is that a good way to evaluate?

Let’s chat!

Legacy code

Is it bad to propose new languages when people are already so experienced with existing ones? When they have so many libraries available? So much code already written?

What else can we use to evaluate if PLs, abstractions, programming systems, programming tools contribute something valuable?

If we won’t eval **usability**, **covering everything**, and if we allow we don’t have to be backwards compatible with all **legacy code**?

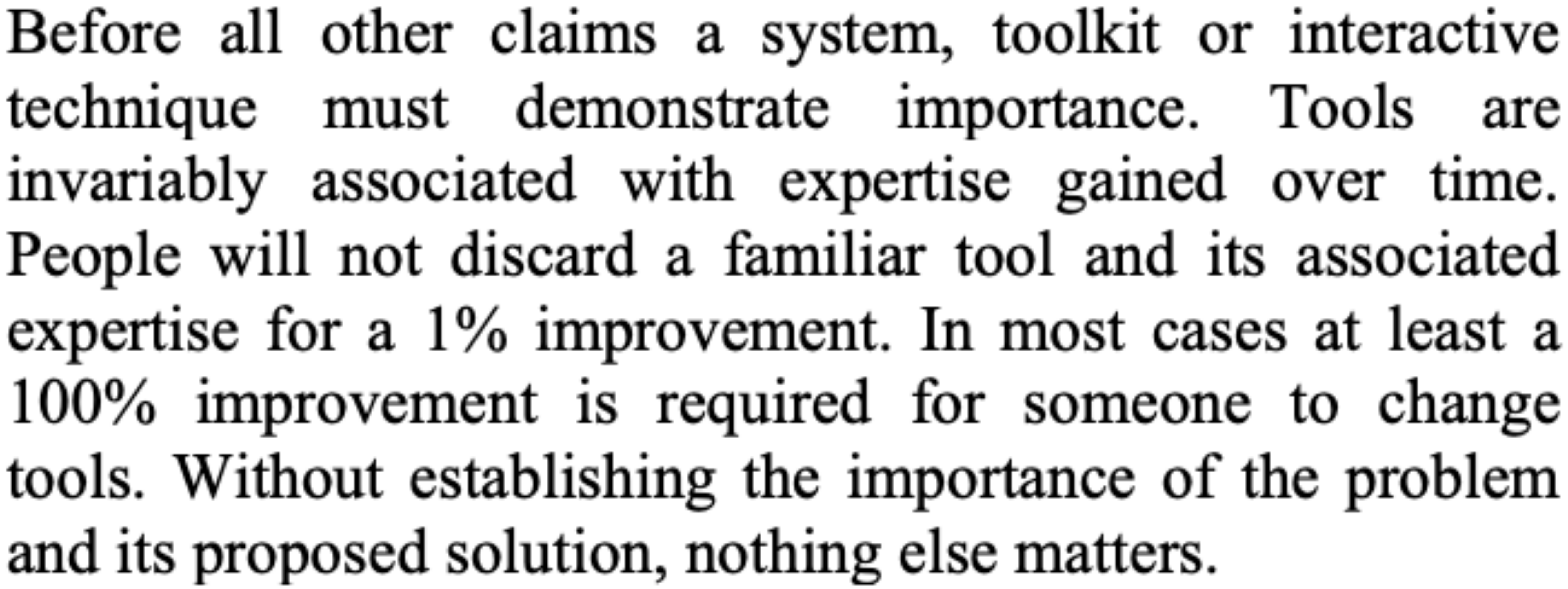
For the next few slides, we’re going to take the reading’s contribution types one at a time.

In your groups, please **brainstorm ways to demonstrate these claims for PL/ Programming Systems contributions**.

I recommend having the reading open in front of you if possible, for inspiration. But I also recommend brainstorming on your own before you refer back to it!

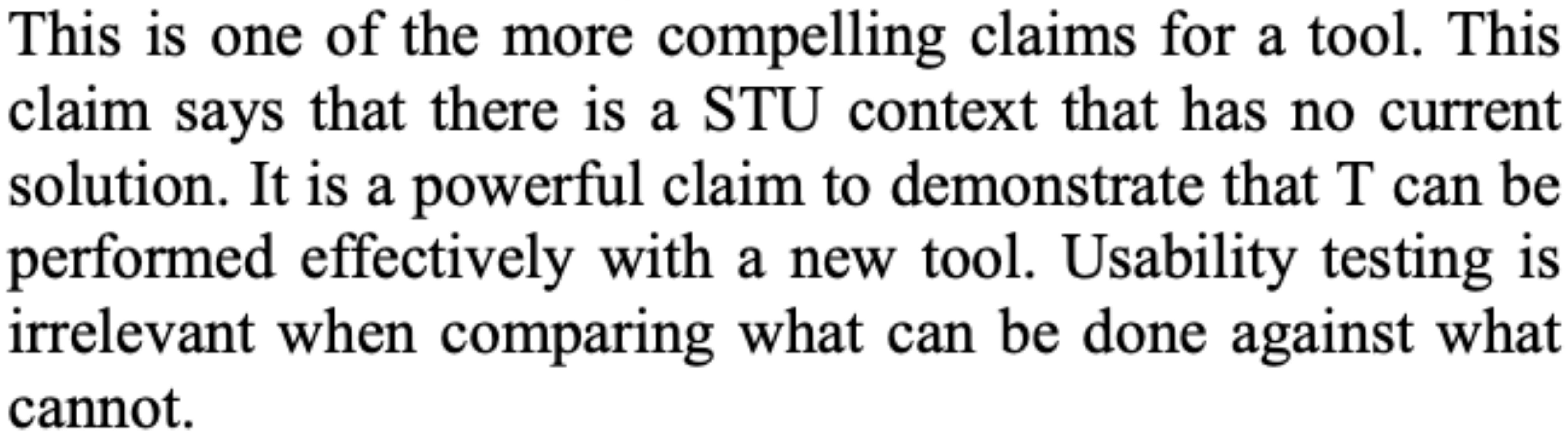
If you struggle to come up with ideas, try making it more concrete. How would you assess this contribution for existing PL or PL-adjacent projects you've seen?

Importance



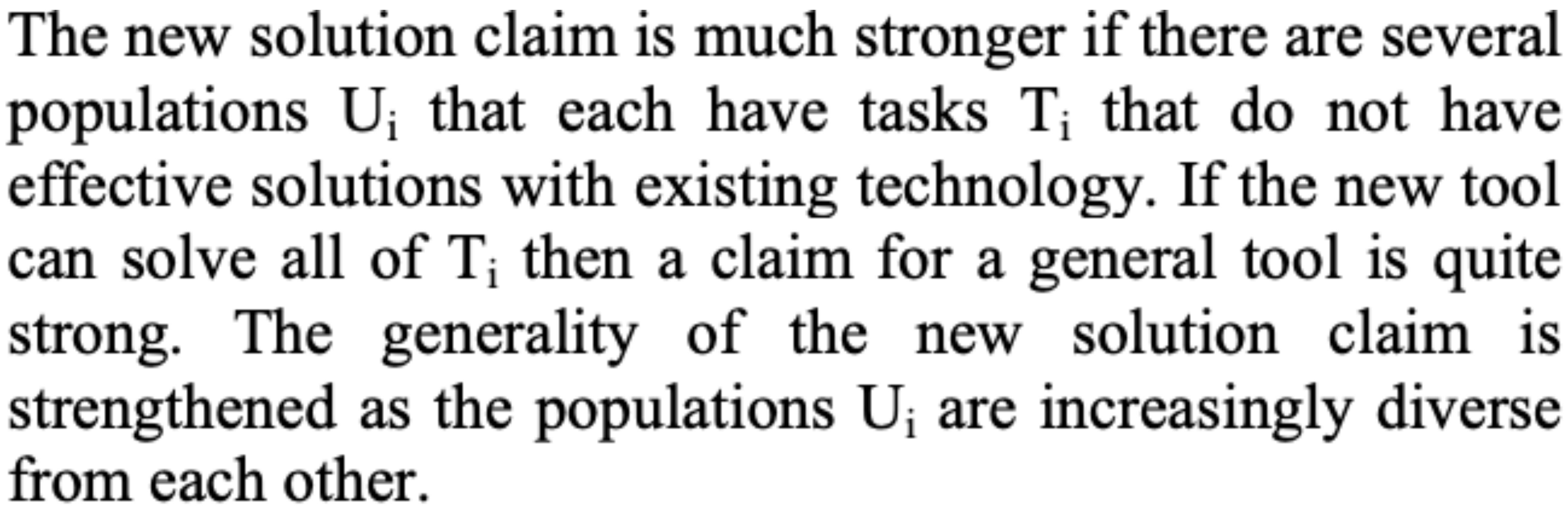
Before all other claims a system, toolkit or interactive technique must demonstrate importance. Tools are invariably associated with expertise gained over time. People will not discard a familiar tool and its associated expertise for a 1% improvement. In most cases at least a 100% improvement is required for someone to change tools. Without establishing the importance of the problem and its proposed solution, nothing else matters.

Problem not previously solved



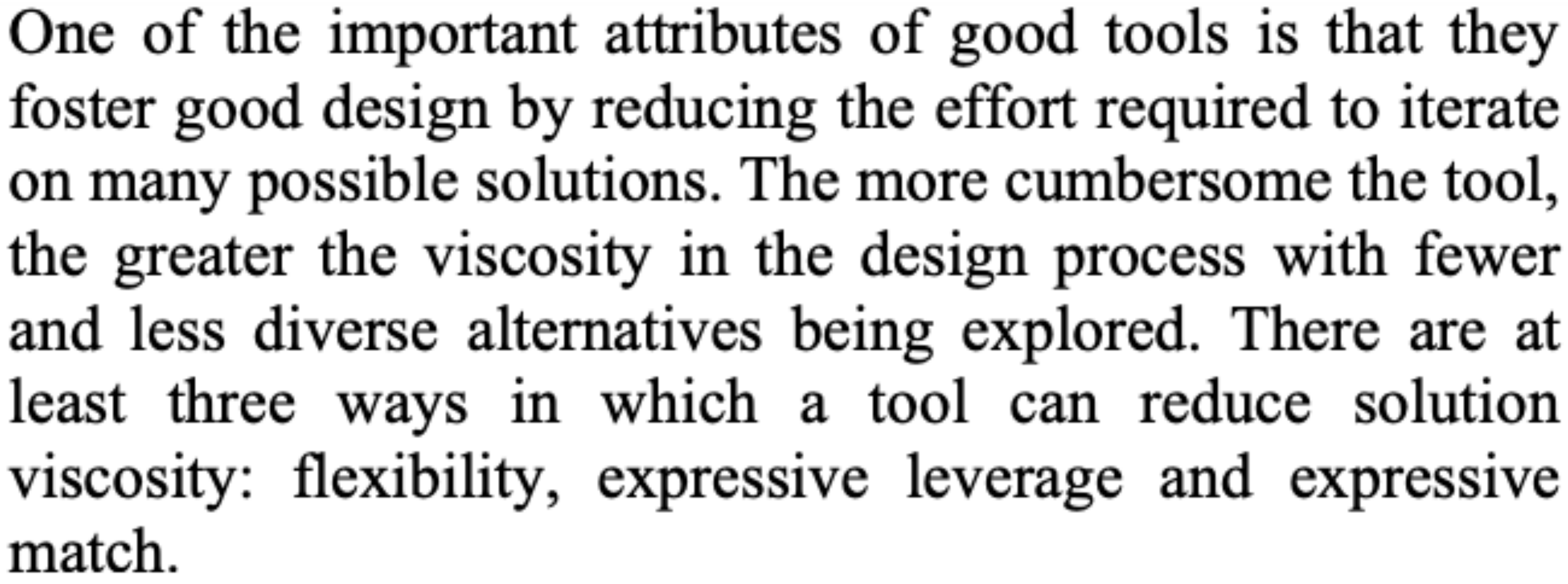
This is one of the more compelling claims for a tool. This claim says that there is a STU context that has no current solution. It is a powerful claim to demonstrate that T can be performed effectively with a new tool. Usability testing is irrelevant when comparing what can be done against what cannot.

Generality



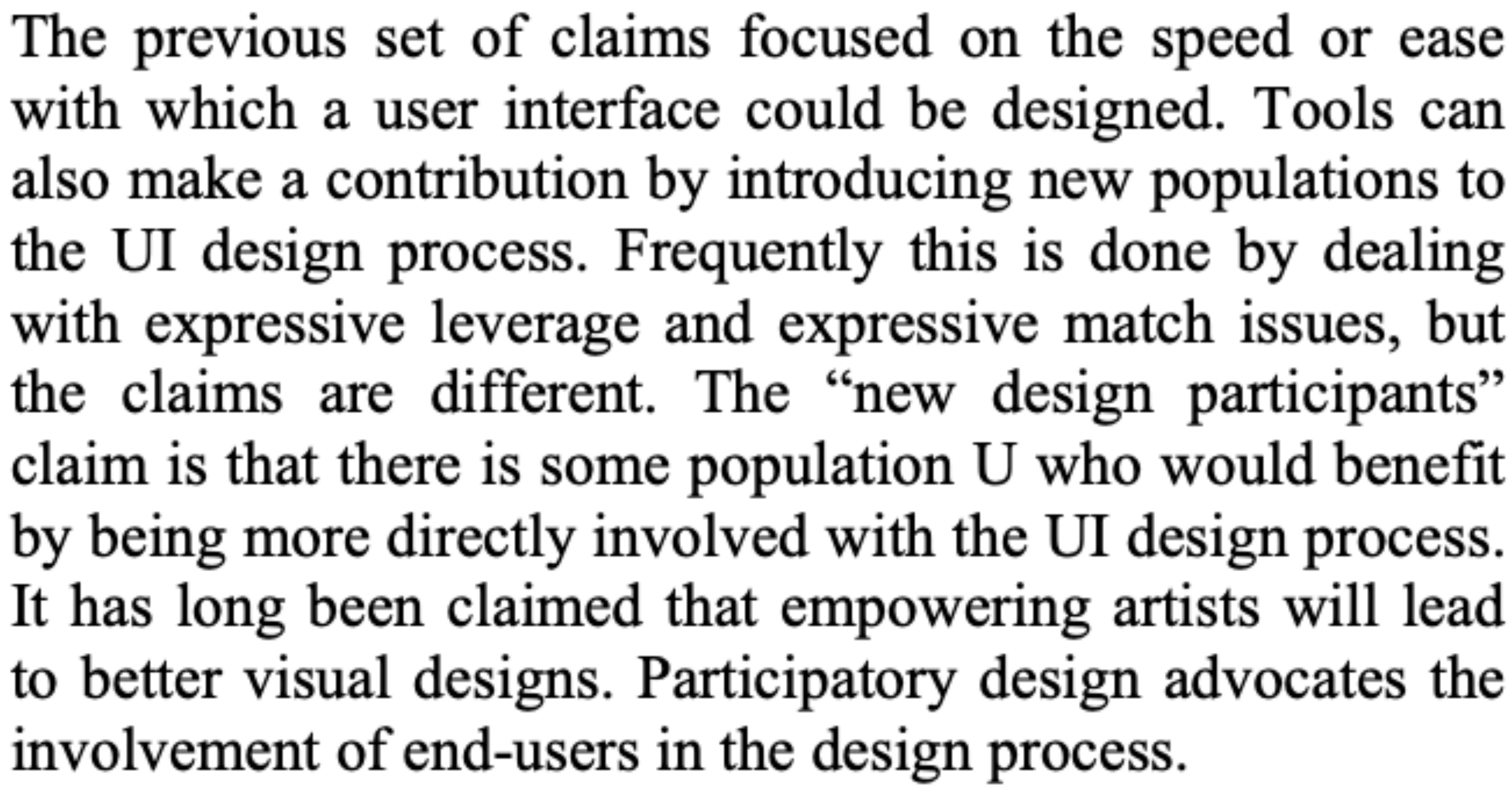
The new solution claim is much stronger if there are several populations U_i that each have tasks T_i that do not have effective solutions with existing technology. If the new tool can solve all of T_i then a claim for a general tool is quite strong. The generality of the new solution claim is strengthened as the populations U_i are increasingly diverse from each other.

Reduce solution viscosity



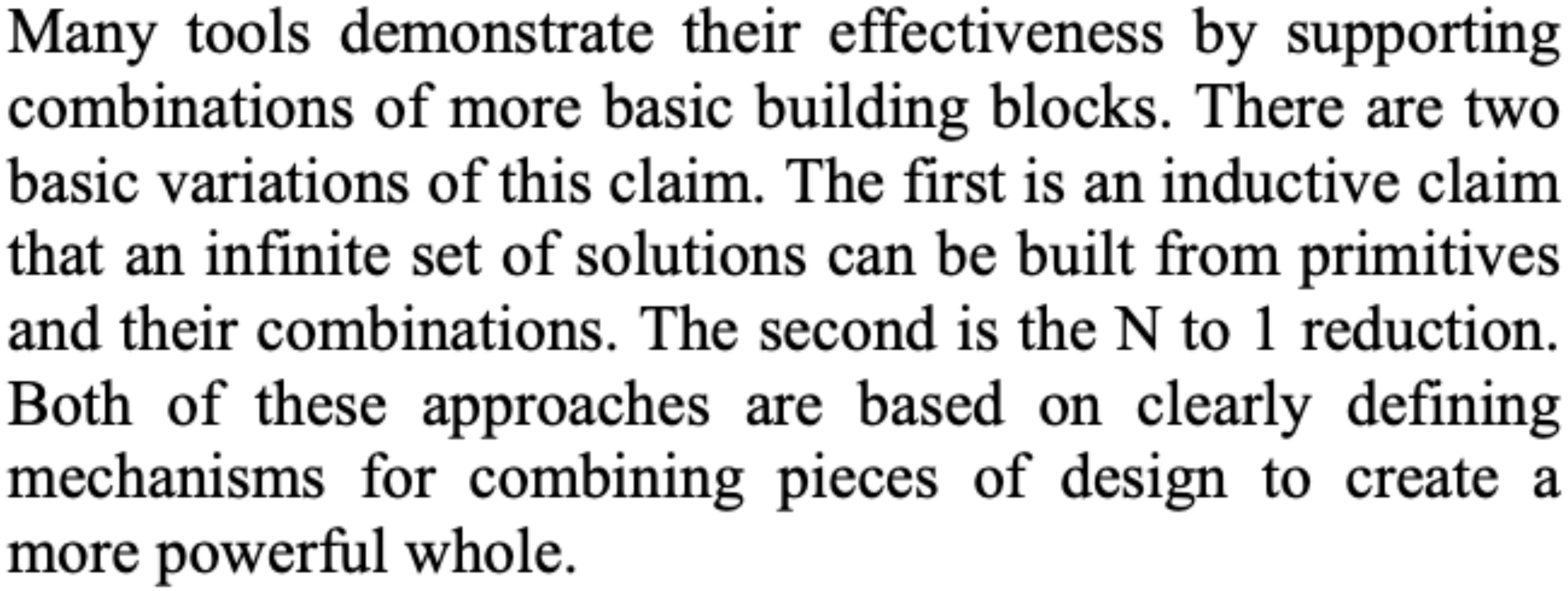
One of the important attributes of good tools is that they foster good design by reducing the effort required to iterate on many possible solutions. The more cumbersome the tool, the greater the viscosity in the design process with fewer and less diverse alternatives being explored. There are at least three ways in which a tool can reduce solution viscosity: flexibility, expressive leverage and expressive match.

Empowering new design participants



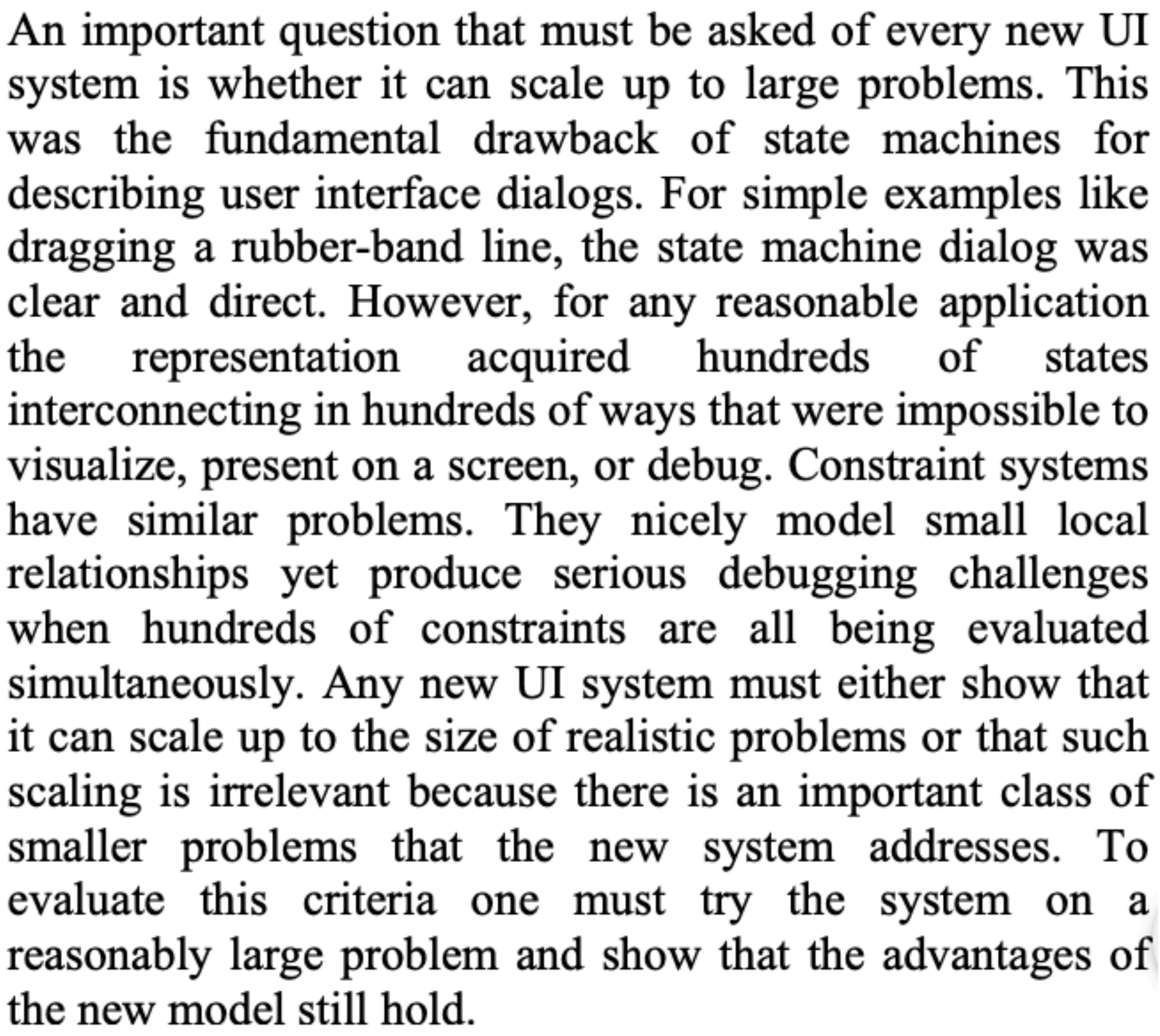
The previous set of claims focused on the speed or ease with which a user interface could be designed. Tools can also make a contribution by introducing new populations to the UI design process. Frequently this is done by dealing with expressive leverage and expressive match issues, but the claims are different. The “new design participants” claim is that there is some population U who would benefit by being more directly involved with the UI design process. It has long been claimed that empowering artists will lead to better visual designs. Participatory design advocates the involvement of end-users in the design process.

Power in combination



Many tools demonstrate their effectiveness by supporting combinations of more basic building blocks. There are two basic variations of this claim. The first is an inductive claim that an infinite set of solutions can be built from primitives and their combinations. The second is the N to 1 reduction. Both of these approaches are based on clearly defining mechanisms for combining pieces of design to create a more powerful whole.

Can it scale up?



An important question that must be asked of every new UI system is whether it can scale up to large problems. This was the fundamental drawback of state machines for describing user interface dialogs. For simple examples like dragging a rubber-band line, the state machine dialog was clear and direct. However, for any reasonable application the representation acquired hundreds of states interconnecting in hundreds of ways that were impossible to visualize, present on a screen, or debug. Constraint systems have similar problems. They nicely model small local relationships yet produce serious debugging challenges when hundreds of constraints are all being evaluated simultaneously. Any new UI system must either show that it can scale up to the size of realistic problems or that such scaling is irrelevant because there is an important class of smaller problems that the new system addresses. To evaluate this criteria one must try the system on a reasonably large problem and show that the advantages of the new model still hold.

What do we get to claim?

- The fact that there are other ways to demonstrate value of PL/Programming Systems contribution doesn’t mean we get to make unsupported usability claims
  - Demonstrating one of these contributions doesn’t mean the tool is usable or that we get to make usability claims without usability eval
- Don’t get to make unsupported claims about these alternative contributions either!
  - But we do get to think creatively about how we evaluate them

So why’d we do this?

- Usability isn’t the only thing we can evaluate.
- Sometimes it’s not practical to evaluate it for PLs.
- …but we have alternatives available! We don’t have to just give up on human factors evaluations.
- The range of options means we have to be thoughtful about our goals, what we want to claim, what we evaluate

Takeaways

- Highly encourage you before designing an evaluation to decide which of these dimensions (or others) about which you want to make claims
  - Sit down with the list, write out the specific claim
  - *Then* design the eval