My vision on research

(don't quote me on the title)

Laurent Vanbever—NSG Group Retreat, Oct 20 2021, Splügen, Switzerland

~~My~~ vision on research

~~My~~ vision on research

Other people's

I happen to agree with

~~My~~ vision on ~~research~~

Other people's

I happen to agree with

~~My~~ vision on ~~research~~

Other people's

I happen to agree with

something else

which very much relates to research

My vision on research

(don't quote me on the title)

My vision on research

(don't quote me on the title)

My vision on research

(don't quote me on the title)

Dieter Rams (1932–)

Industrial Designer (Braun)

My vision on research

(don't quote me on the title)

My vision on research

(don't quote me on the title)

Christoph Niemann (1970–)

Illustrator

In my office…

Dieter Rams

Ten Commandments of Design

Christoph Niemann

Creative Process

Dieter Rams

Ten Commandments of Design

Christoph Niemann

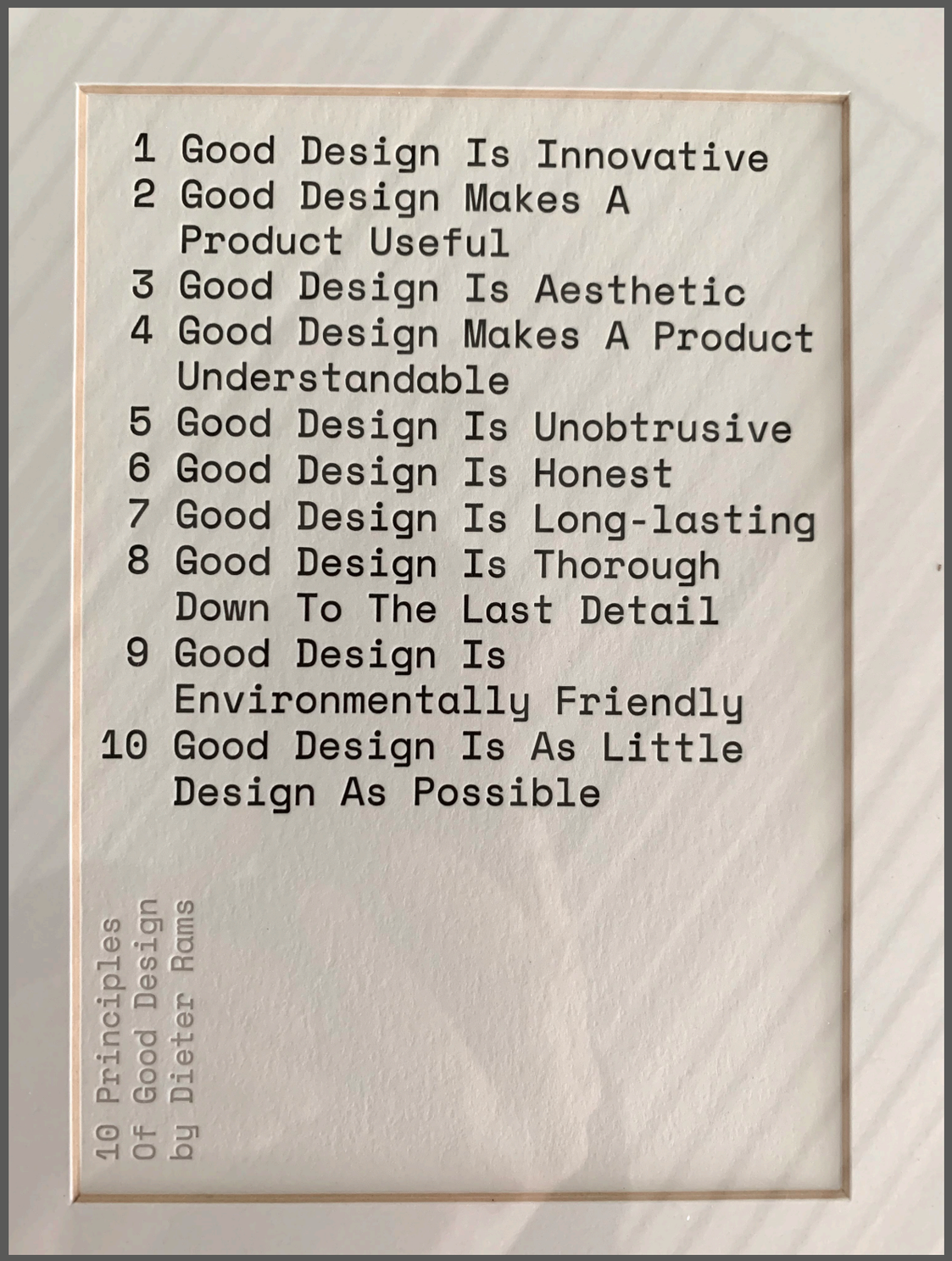
Creative Process

What is good research?

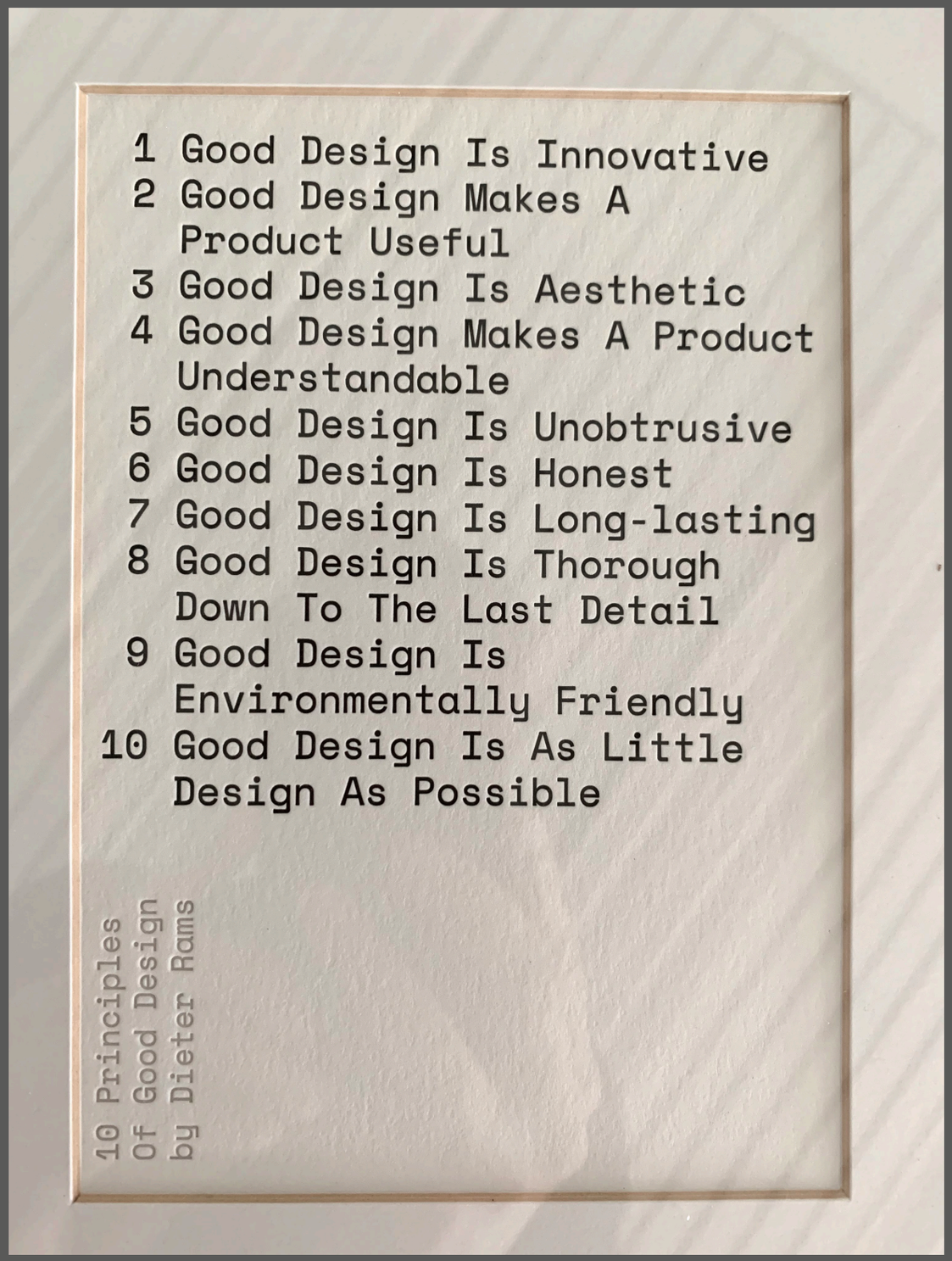
1 Good Design Is Innovative
2 Good Design Makes A Product Useful
3 Good Design Is Aesthetic
4 Good Design Makes A Product Understandable
5 Good Design Is Unobtrusive
6 Good Design Is Honest
7 Good Design Is Long-lasting
8 Good Design Is Thorough Down To The Last Detail
9 Good Design Is Environmentally Friendly
10 Good Design Is As Little Design As Possible

10 Principles
Of Good Design
by Dieter Rams

s/Design/Research

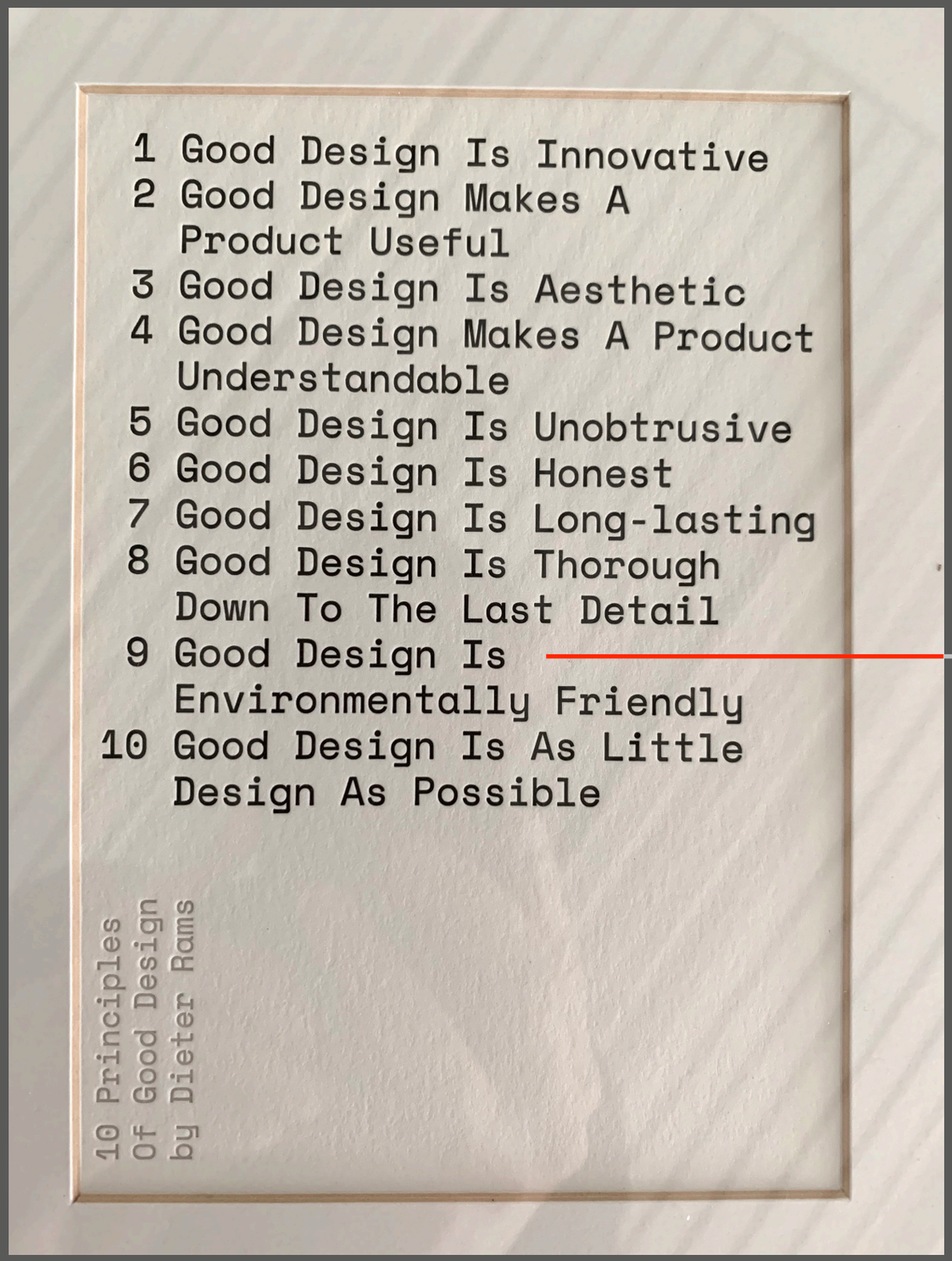


s/Design/Research



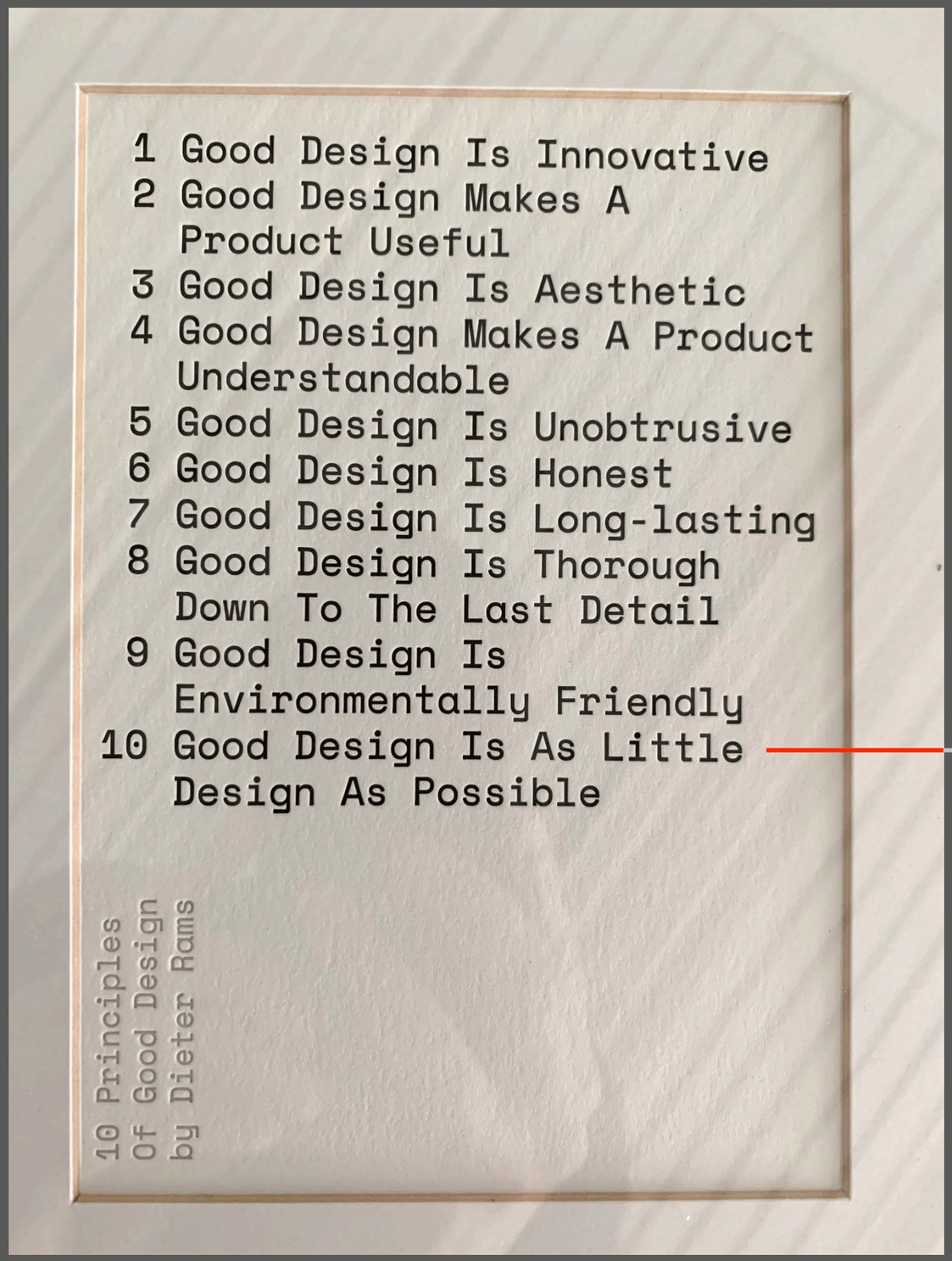
perfect match

s/Design/Research



don't directly apply (thus far)

s/Design/Research



really, the opposite (for us)

So you want to be an (independent) researcher?

And publish large

A big group. Five SIGCOMM a year. You're in charge.

Comin' up in the world.

—Cypress Hill

Graduate school is about turning you into independent researchers

Graduate school is about turning you into

independent researchers

I'm working myself "out of the job"

Graduate school is about turning you into

independent researchers

I'm working myself "out of the job"

this is typically a **3-steps process**

step #1

step #1 Given a question

step #1 Given a question and an approach

step #1

Given a question and an approach

Deliver a solution

#1 Given a question and an approach

Deliver a solution

step #2

#1 Given a question and an approach

Deliver a solution

step #2 Given a question,

#1 Given a question and an approach

Deliver a solution

step #2 Given a question,

Develop an approach

#1 Given a question and an approach

Deliver a solution

step #2 Given a question,

Develop an approach

Deliver a solution

#1 Given a question and an approach

Deliver a solution

#2 Given a question,

Develop an approach

Deliver a solution

step #3

#1 Given a question and an approach

Deliver a solution

#2 Given a question,

Develop an approach

Deliver a solution

step #3 Identify a question

#1 Given a question and an approach

Deliver a solution

#2 Given a question,

Develop an approach

Deliver a solution

step #3 Identify a question

Develop an approach

#1 Given a question and an approach

Deliver a solution

#2 Given a question,

Develop an approach

Deliver a solution

step #3 Identify a question

Develop an approach

Deliver a solution

#1 Given a question and an approach

Deliver a solution

junior student

#2 Given a question,

Develop an approach

Deliver a solution

senior student

#3 Identify a question

Develop an approach

Deliver a solution

independent researcher

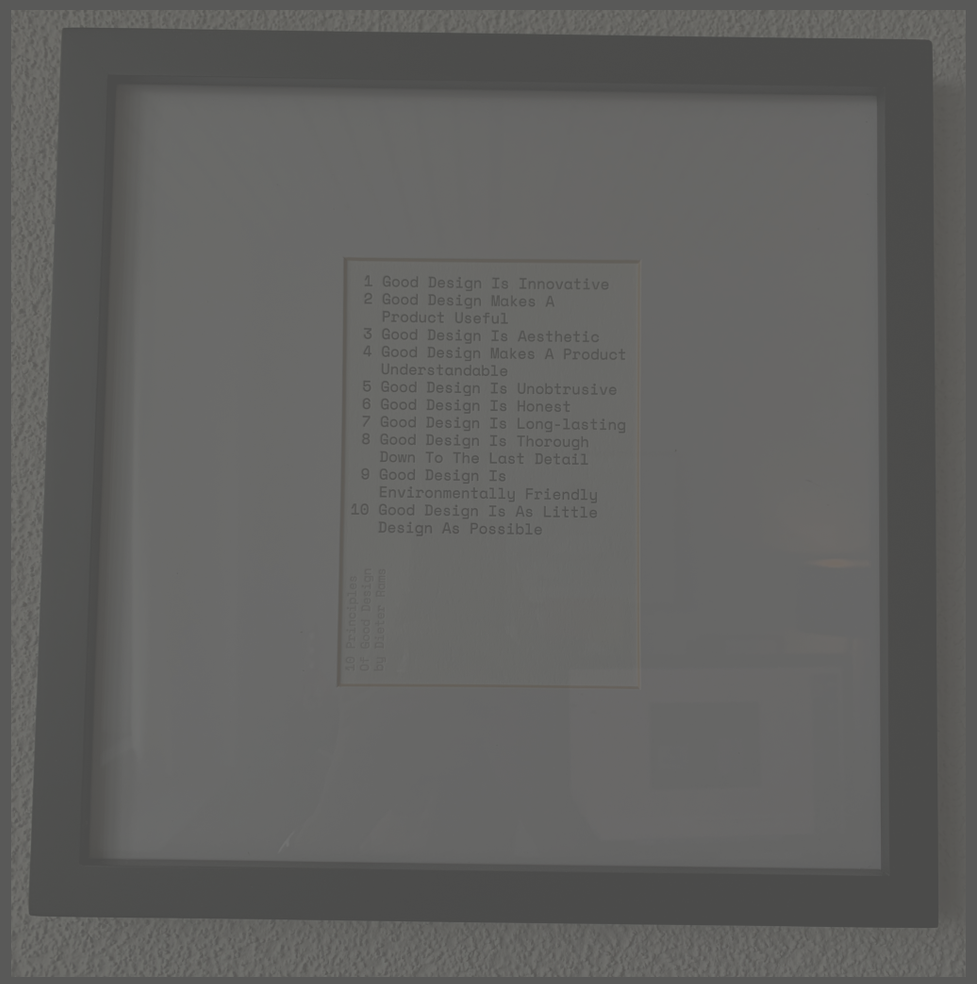
In practice, each step tends to match to one paper

but it's *not* mandatory

It's *really* about

being able to work as an independent researcher

#3	Identify	a question	**independent researcher**
	Develop	an approach	
	Deliver	a solution	



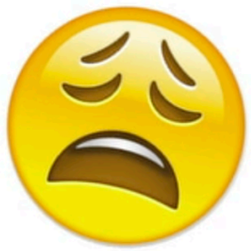
Dieter Rams

Ten Commandments of Design

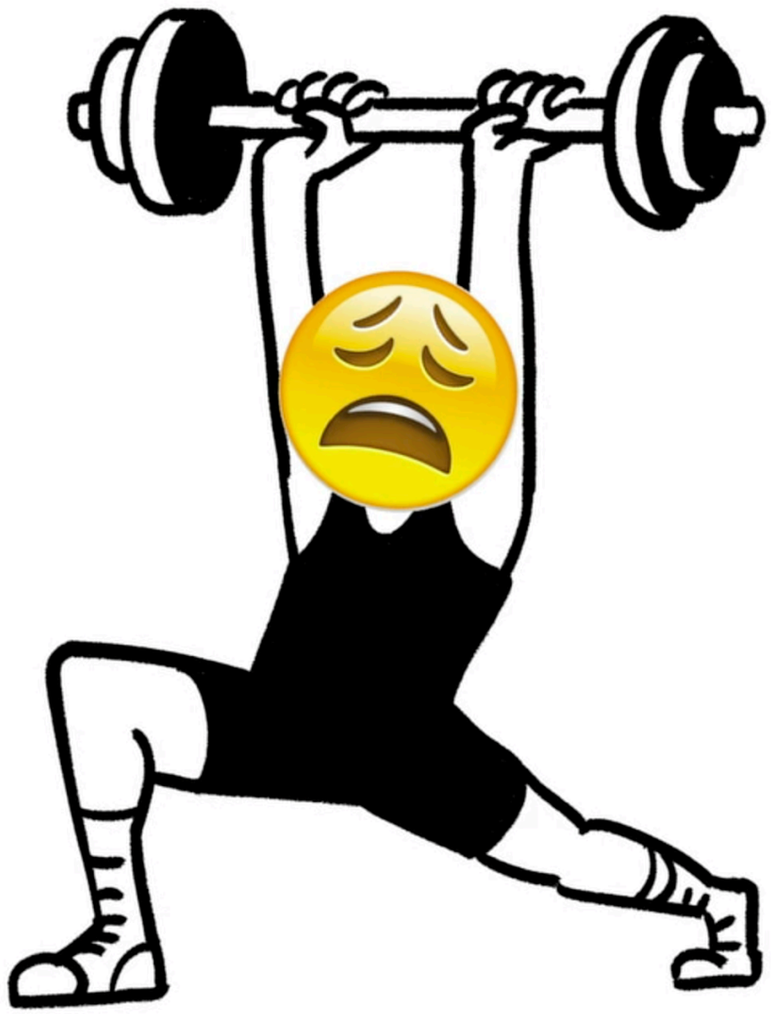
Christoph Niemann

Creative Process

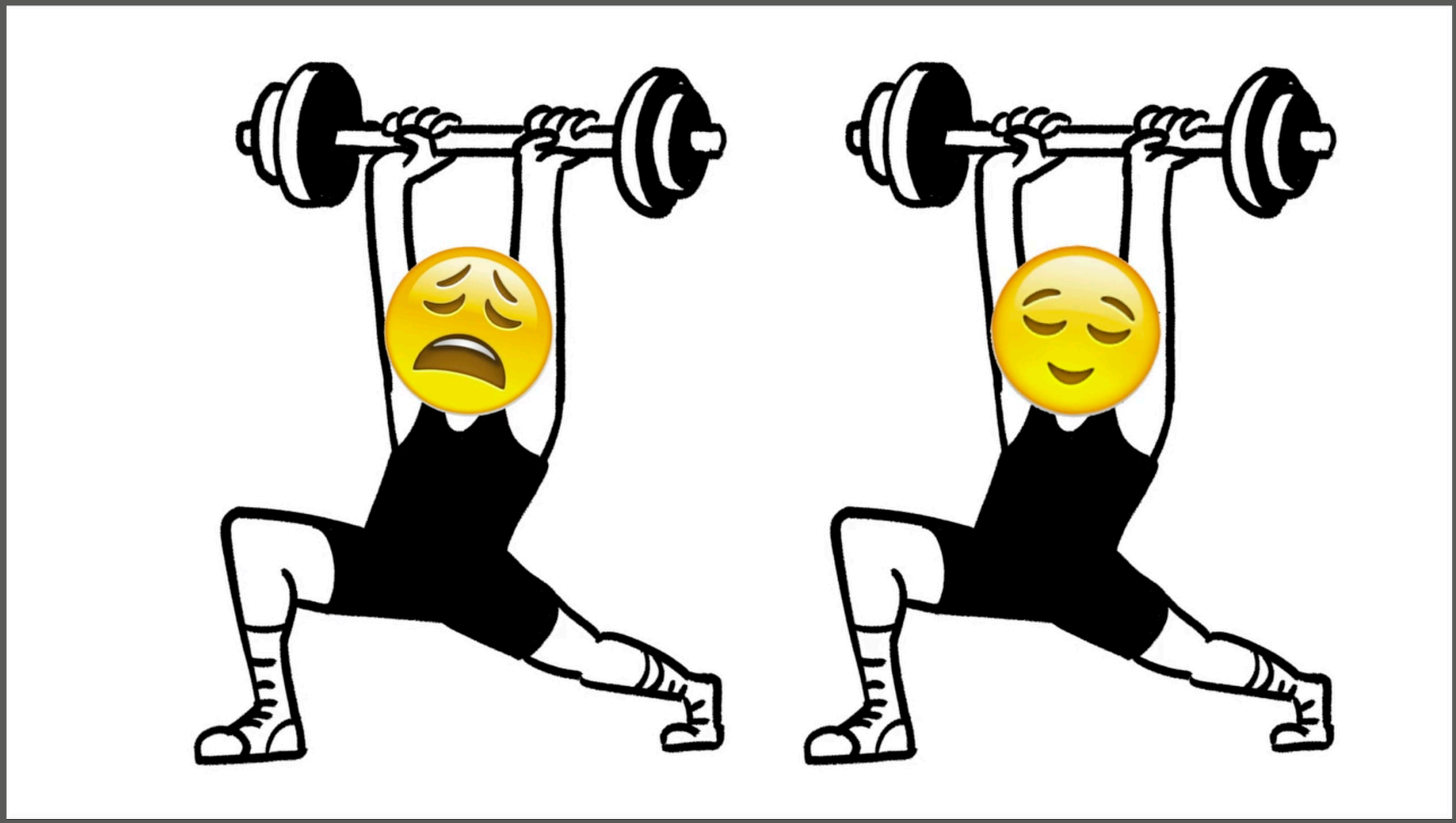
Source: New Yorker



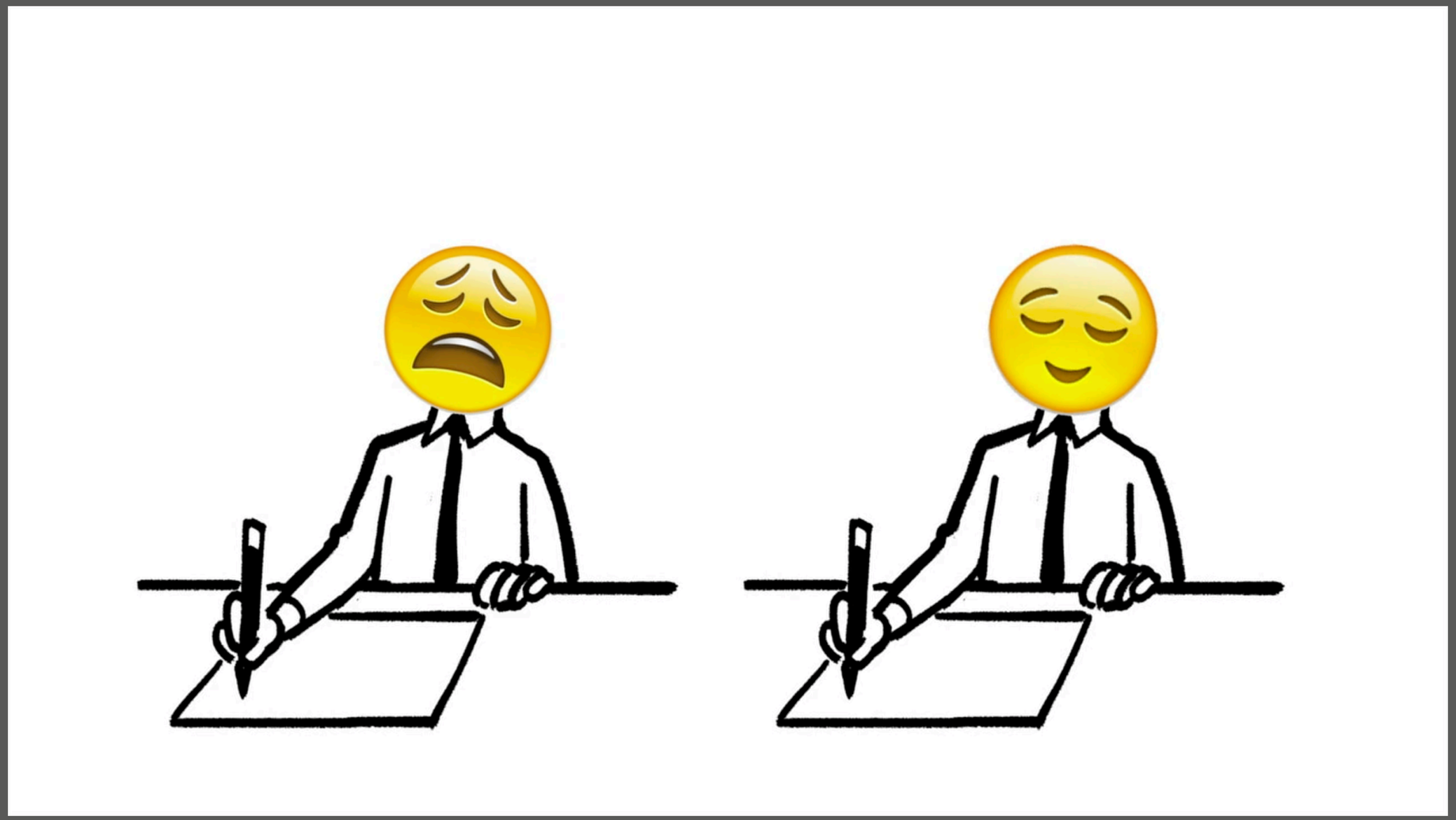
Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)



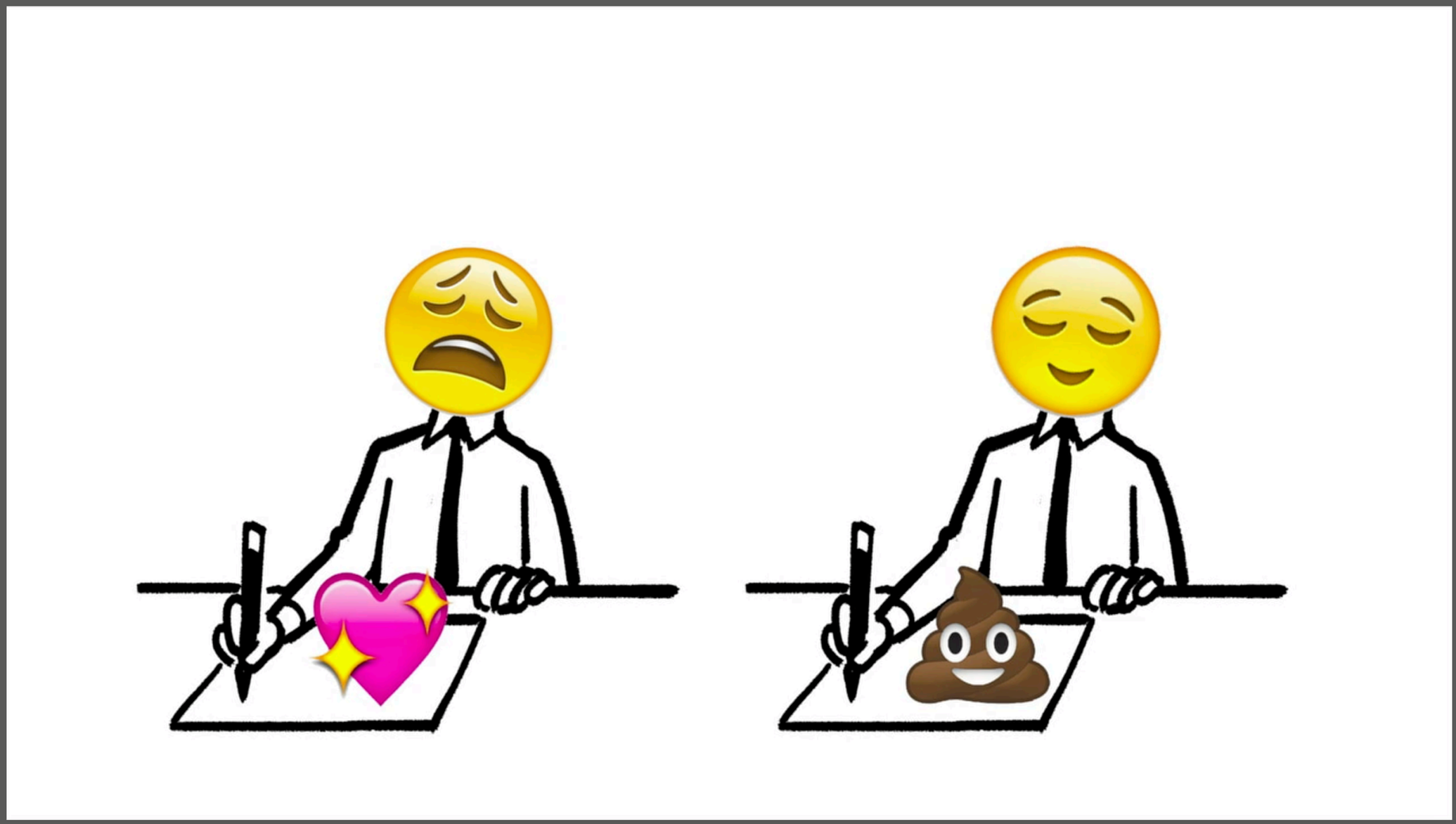
Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)

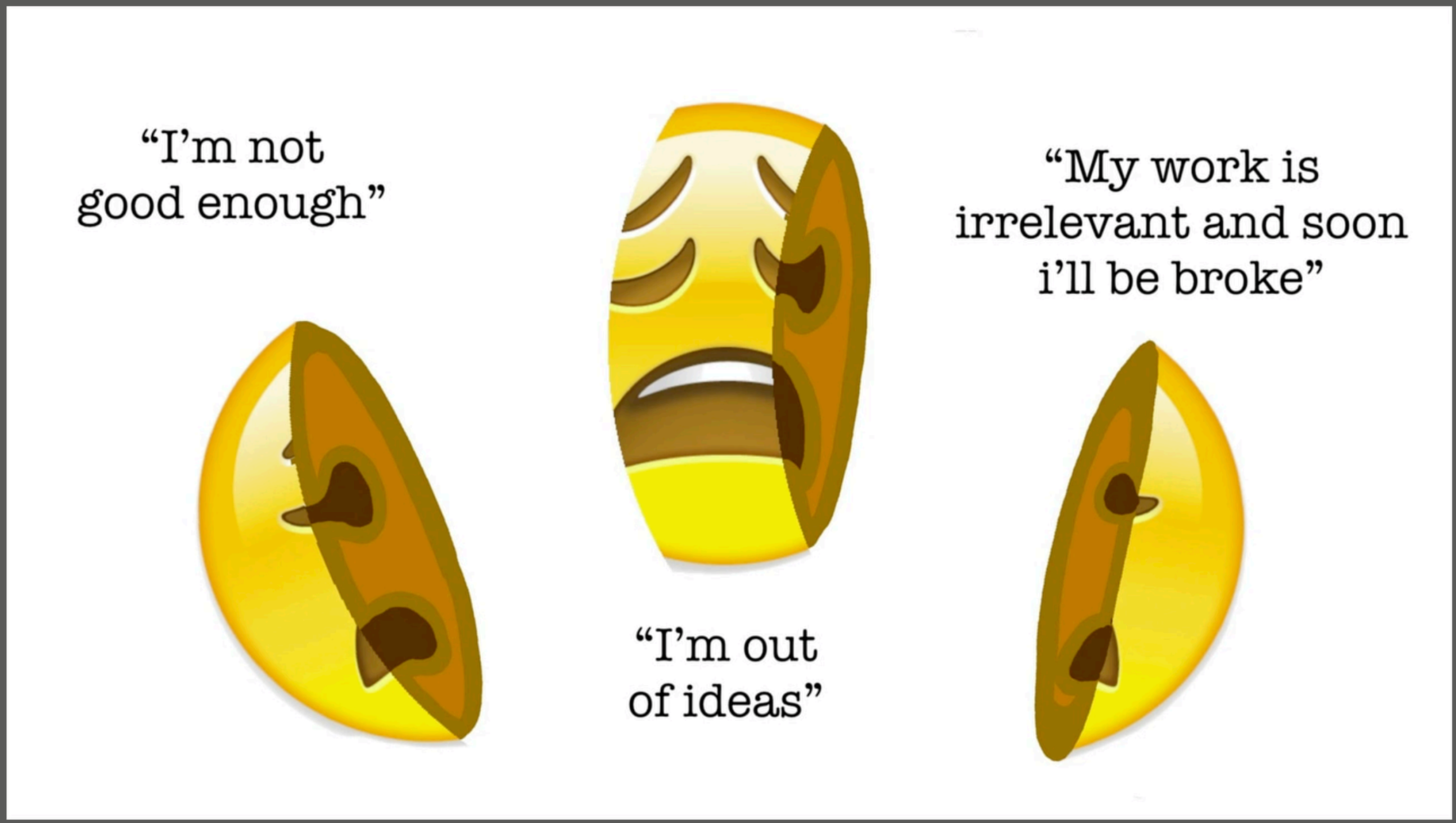


Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)


37/50
NIEMANN 2017



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)

Problem 1

I'm not good enough

Problem 1

I'm not good enough

Solution 1

Relax! Don't be so hard on yourself!

Problem 1

I'm not good enough

Solution 1

~~Relax! Don't be so hard on yourself!~~

Problem 1

I'm not good enough

Solution 1

~~Relax! Don't be so hard on yourself!~~

Solution 2

Practice to become better

Doing good research is not something you're good at...

it's something you become good at

Doing good research is not something you're good at...
it's something you become good at

It requires *exactly*...

10,000 hours

~5 years (40 hours/year)

Trust me...

I've come a long way since my first "research" work

Towards validated network configurations with NCGuard

Laurent Vanbever, Grégory Pardoen, Olivier Bonaventure
Université catholique de Louvain, Belgium

***Abstract*— Today, most IP networks are still configured manually on a router-by-router basis. This is error-prone and often leads to misconfiguration. In this paper, we describe the Network Configuration Safeguard (`NCGuard`), a tool that allows the network architect to apply a safer methodology. The first step is to define his design rules. Based on a survey of the networking literature, we classify the most common types of rules in three main patterns: presence, uniqueness and symmetry and provide several examples. The second step is to write a high-level representation of his network. The third step is to validate the network representation and generate the configuration of each router. This last step is performed automatically by our prototype. Finally, we describe our prototype and apply it to the Abilene network.**

I. Introduction

Managing and configuring the devices that compose an IP network is a complex, costly and error-prone task [1]. Network operators must ensure consistency among neighboring routers while facing complex configuration languages and frequent changes in the network [2], [3]. Furthermore, a typical network may contain hundreds of devices from different vendors running different operating systems with their own configuration language.

In many, if not most, networks, the state of the art methodology used by network engineers to configure their routers and switches is very simple [4]. A wiki or a set of text files contain the network documentation, policies and small configuration templates for the most common tasks [2]. When a router's configuration must be changed, the network engineers often have compared the current way of configuring networks to *writing a distributed program in assembly language* [8].

In this paper, we describe the Network Configuration Safeguard (`NCGuard`). `NCGuard` is a first, but important, step towards the utilization of software engineering techniques to produce network configurations that can be validated. `NCGuard` encourages the network architect to first specify formally the objectives of his network. Such a formalization is used by several software engineering techniques such as *design by contract* [9] for example. These objectives are defined as a set of rules that must be met by the configuration. Then, the network architect writes a high-level representation of his network. `NCGuard` then validates the configuration automatically based on the rules defined by the architect and generates the routers' configurations in their respective configuration languages.

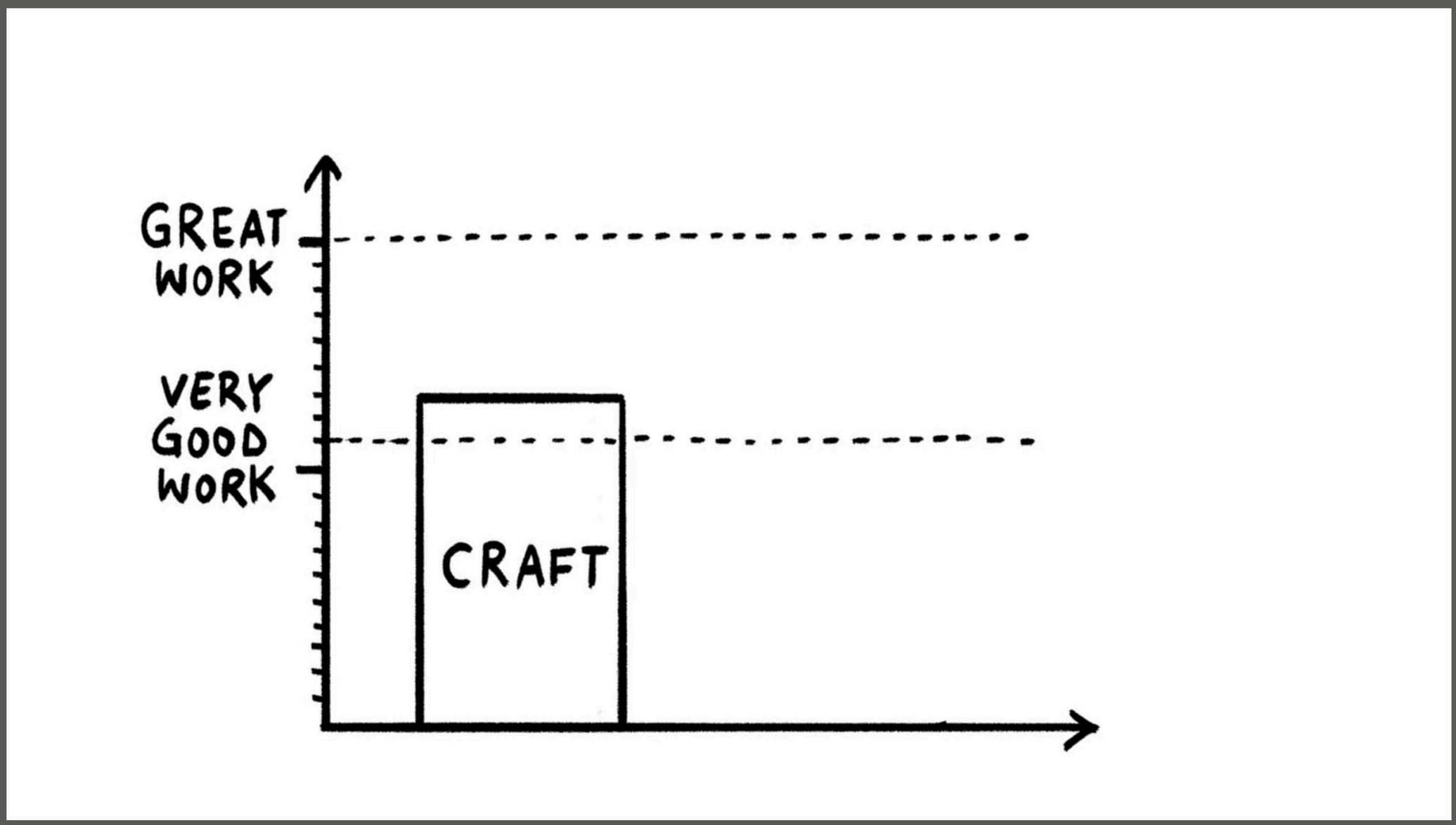
This paper is organized as follows. We first describe in Section II how a network architect can specify his objectives. Section III details the design of `NCGuard`. Section IV uses `NCGuard` to produce the configuration of the Abilene network and Section V discusses related work.

II. Configuration objectives

Network architects usually have two different types of configuration objectives. High-level objectives are design decisions made by the architect about the organization of his network. Some examples of high-level objectives are: ensure that the iBGP sessions distribute interdomain routes to all BGP routers, ensure that all intradomain routes are distributed by

With sheer craft,

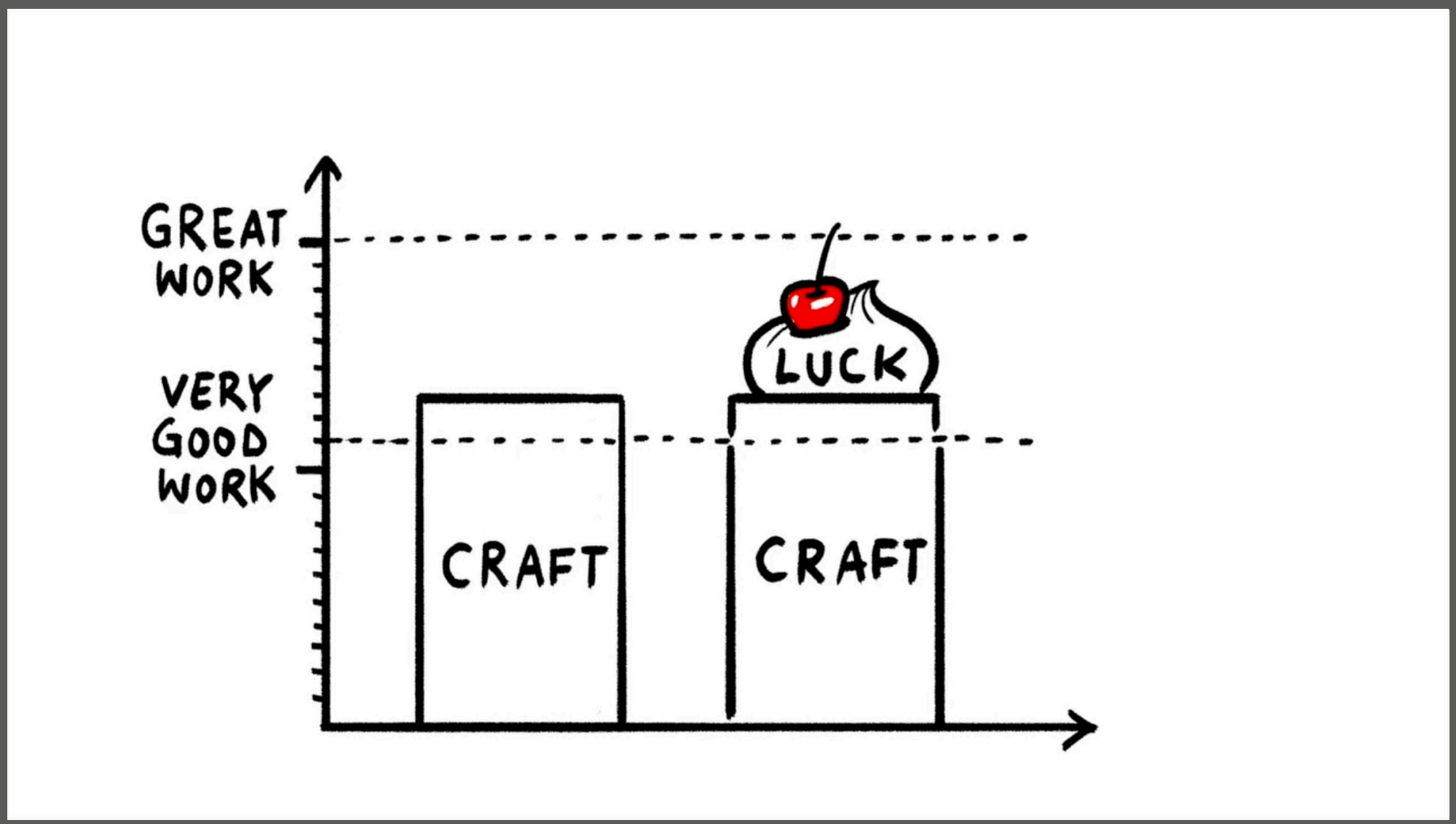
you can *reliably* create (very) good work



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)

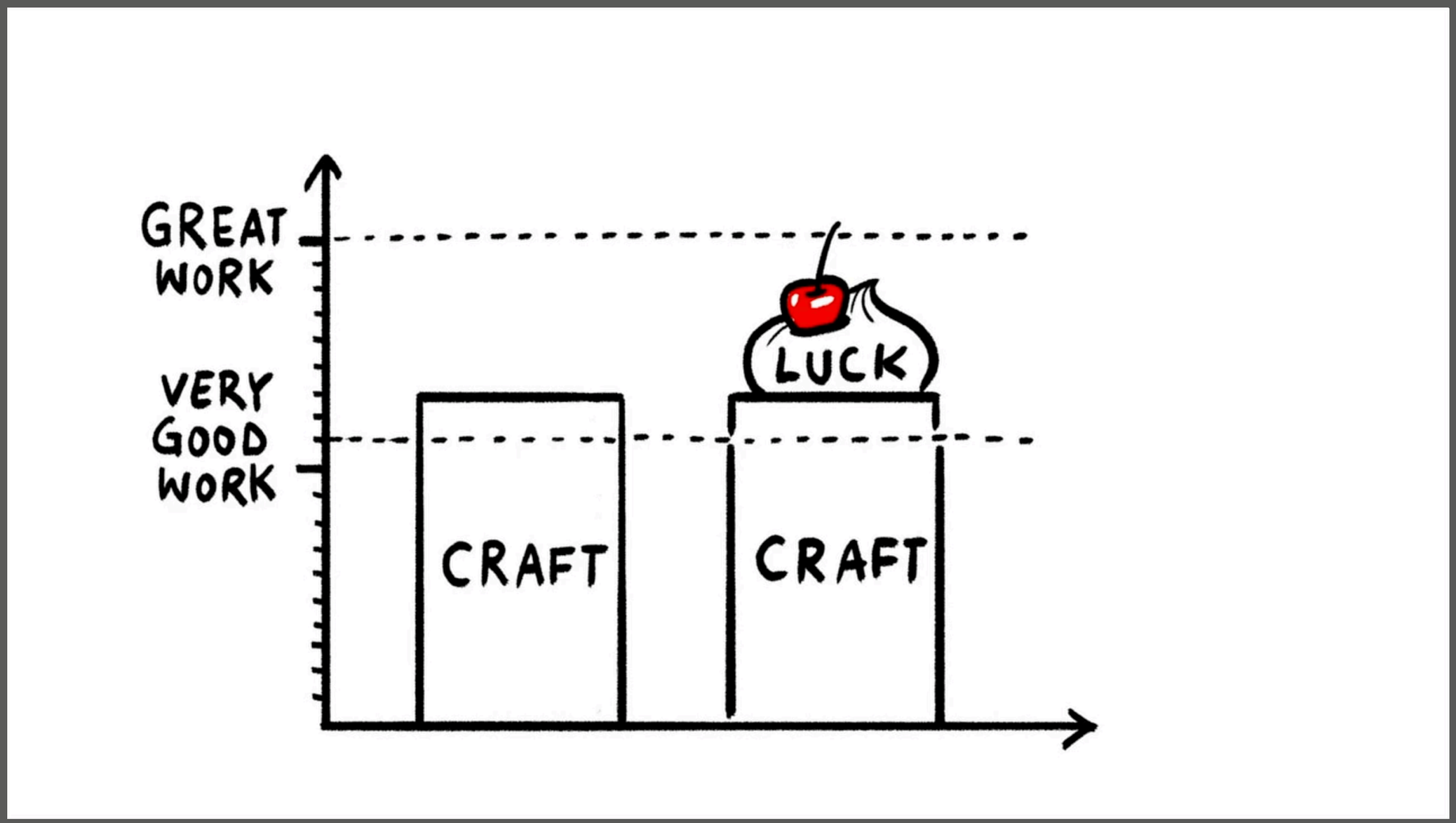
For doing great work though…

you also need… a ***lot*** of luck



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)

You can and should only aspire to do very good work, great work happens or doesn't happen



Source: Christoph Niemann: How to Overcome the 3 Fears Every Creative Faces (YouTube)

You can only aspire to do very good work,

great work happens or doesn't happen

you "only" need very good work to graduate

or get faculty position

Of course, the risk is always that you start mastering a craft that no ones really care about

Of course, the risk is always that you start mastering a craft that no ones really care about

I'm here to avoid that (as much as possible)

Problem 2

My work is predictable and soon I'll be broke

Problem 2

My work is predictable and soon I'll be broke

Solution

Worry, doubt, and agonize

You've to be your harshest critic

(this is hard, introspective work)

Is my work shallow?

Am I taking enough creative risks?

Am I in touch with what's happening out there?

Of course, getting external feedback

but need to be taken with a grain of 

Problem 3

I'm out of ideas!

Problem 3

I'm out of ideas!

Solution

Create.

Continously nurture a list of problems, try/learn things, be on constant lookout for "new clicks"

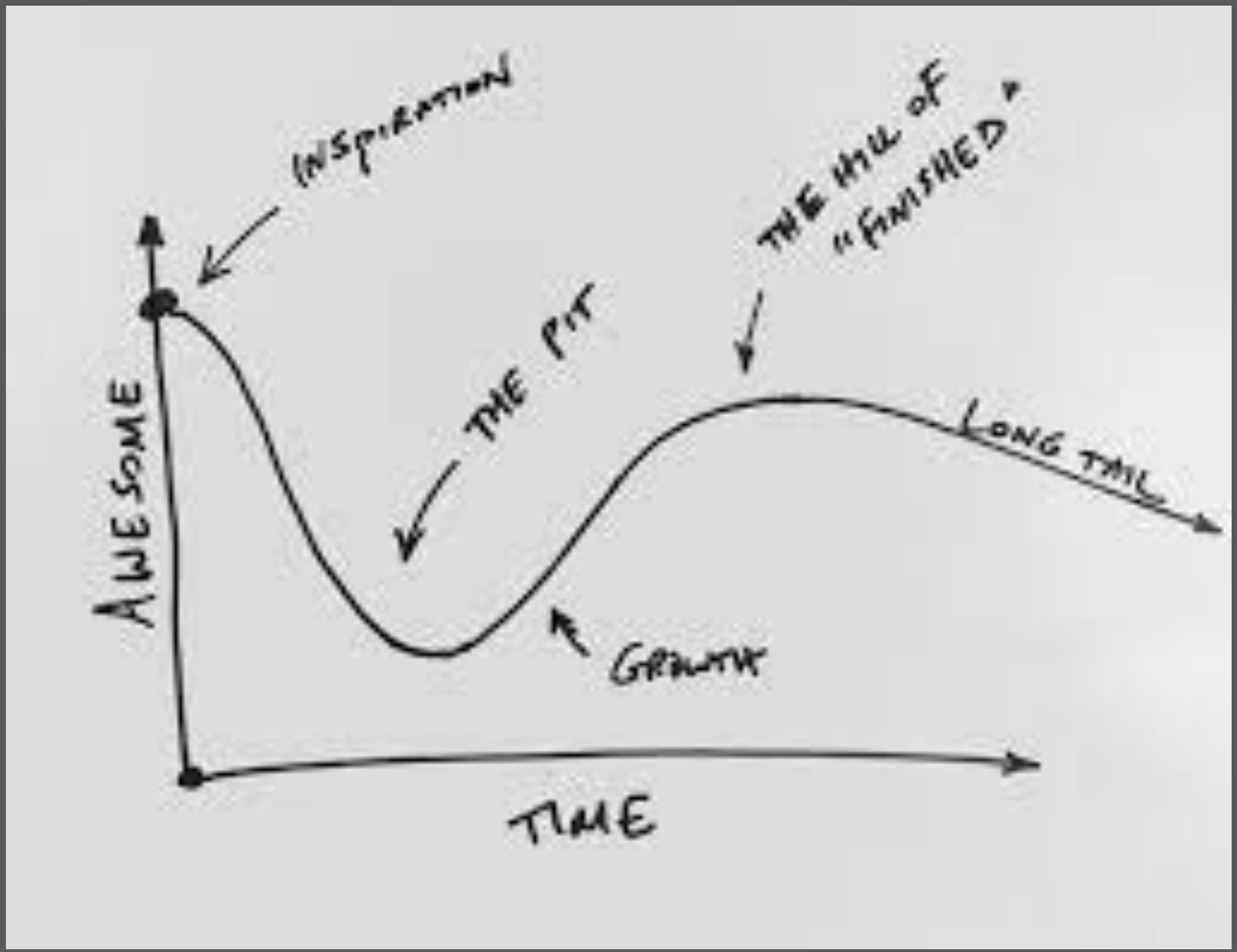
You MUST dedicate time for that

this is your creative life insurance

Continously nurture a list of problems, try/learn things,

be on constant lookout for "new clicks"

from there, your job is to bring the idea to fruition
without killing it. This is **very hard**.


INSPIRATION
THE PIT
THE HILL OF "FINISHED"
LONG TAIL
GROWTH
AWESOME
TIME

The ninety-ninety rule
of paper writing

Writing the first 90 percent of the paper accounts for
the first 90 percent of the writing time.

Writing the remaining 10 percent accounts for
the other 90 percent of the development time.

—Tom Cargill, Bell Labs
revisited by yours truly

Using Routers to Build Logic Circuits: How Powerful is BGP?

Marco Chiesa* Luca Cittadini* Giuseppe Di Battista* Laurent Vanbever* Stefano Vissicchio†
*Roma Tre University *Princeton University †Université catholique de Louvain
*{chiesa,ratm,gdb}@dia.uniroma3.it *vanbever@cs.princeton.edu †stefano.vissicchio@uclouvain.be

***Abstract*—Because of its practical relevance, the Border Gateway Protocol (BGP) has been the target of a huge research effort since more than a decade. In particular, many contributions aimed at characterizing the computational complexity of BGP-related problems. In this paper, we answer computational complexity questions by unveiling a fundamental mapping between BGP configurations and logic circuits. Namely, we describe simple networks containing routers with elementary BGP configurations that simulate logic gates, clocks, and flip-flops, and we show how to interconnect them to simulate arbitrary logic circuits. We then investigate the implications of such a mapping on the feasibility of solving BGP fundamental problems, and prove that, under realistic assumptions, *BGP has the same computing power as a Turing Machine*. We also investigate the impact of restrictions on the expressiveness of BGP policies and route propagation (e.g., route propagation rules in iBGP and Local Transit Policies in eBGP) and the impact of different message timing models. Finally, we show that the mapping is not limited to BGP and can be applied to generic routing protocols that use several metrics.**

I. Introduction and Related Work

The Border Gateway Protocol (BGP) [1] is the de-facto routing protocol that regulates inter-domain routing. BGP comes in two flavors: external BGP (eBGP) and internal BGP (iBGP). eBGP is used to exchange reachability information between neighboring networks or Autonomous Systems (AS), while iBGP is used to distribute externally-learned routes within an AS.

BGP enables each AS to apply routing policies in complete autonomy, i.e., enabling each AS to fully control the routes that it accepts, prefers, and propagates to its neighboring ASes. While such a rich policy expressiveness can support complex business relationships, it can also cause routing and forwarding anomalies both in eBGP [2] and iBGP [3] configurations.

Because of its practical relevance for Internet operation and its lack of correctness guarantees, BGP has been the focus of many research and industrial efforts in the last 15 years. Results of such an effort encompass formal analyses of the protocol (*e.g.*, [2], [3]), experimental measurements of disruptions due to BGP (*e.g.*, [4], [5]), proposal of configuration guidelines (*e.g.*, [6]) and of protocol modifications (*e.g.*, [7]), and practical approaches to check a given configuration for correctness (*e.g.*, [8], [9]). However, all previous studies missed a fundamental analogy: Basic BGP configurations can encode elementary logic gates but also, memory and clock components. As such, BGP is powerful enough to encode logic circuits of arbitrary complexity, as we show in Section II.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

We build this mapping assuming a simplified model for BGP routing policies which does not include advanced BGP features like MED or conditional advertisement.

In this paper, we investigate the theoretical consequences of the existence of such a mapping between BGP configurations and logic circuits. We make the following four contributions.

First, we leverage the mapping to characterize the computational complexity of several routing problems in a "bounded" asynchronous model. Contrary to previous works on BGP complexity, in this model each network link is associated with a network delay bounded between finite minimum and maximum values. This effectively imposes a partial order on the exchange of BGP updates. Previous lower bounds for BGP related problems have been proved in models that allow BGP messages to be arbitrarily (even if not indefinitely) delayed [2], [3], [10], [11], [12], [13], [14]. Moreover, the rest of the literature studied BGP from a game-theoretic perspective, where ASes act as players and BGP policies as players' strategies. Messages between players are still allowed to be arbitrarily (even if not indefinitely) delayed. However, these approaches fail to capture specific BGP features either in the game (*e.g.*, by assuming that routers can directly receive routes from non-neighbors [15]) or in the strategies (*e.g.*, by considering impossible strategies in BGP [16], [17]). In Section III, we show that *BGP configurations can simulate arbitrary Turing Machines* in the considered bounded asynchronous model. Two implications derive from this observation. First, policy-based protocols like BGP intrinsically have the same computational power of Turing Machines, even when simple policies are considered. Second, it enables us to assess the computational intractability of BGP routing problems, like routing convergence and correct route propagation.

Second, in Section IV, we use the mapping to investigate the impact of policy restrictions on the complexity of BGP problems. We analyze both iBGP networks and eBGP policy configuration paradigms like the well-known Gao-Rexford conditions [6] and the widely used Local Transit Policies [18]. Also, we discuss the extent to which the mapping holds when other message timings are considered.

Third, in Section V, we show that our methodology can be applied in a routing framework that is different from BGP, and we investigate how difficult the analysis of a generic routing protocol using several metrics is.

Finally, we prove that our approach can be used in several message timing models in Section VI. In particular, we show

Central Control Over Distributed Routing

http://fibbing.net

Stefano Vissicchio*, Olivier Tilmans*, Laurent Vanbever†, Jennifer Rexford‡
* Université catholique de Louvain, † ETH Zurich, ‡ Princeton University
* name.surname@uclouvain.be, †lvanbever@ethz.ch, ‡jrex@cs.princeton.edu

ABSTRACT

Centralizing routing decisions offers tremendous flexibility, but sacrifices the robustness of distributed protocols. In this paper, we present *Fibbing*, an architecture that achieves both flexibility and robustness through central control over distributed routing. Fibbing introduces fake nodes and links into an underlying link-state routing protocol, so that routers compute their own forwarding tables based on the augmented topology. Fibbing is expressive, and readily supports flexible load balancing, traffic engineering, and backup routes. Based on high-level forwarding requirements, the Fibbing controller computes a compact augmented topology and injects the fake components through standard routing-protocol messages. Fibbing works with any unmodified commercial routers speaking OSPF. Our experiments also show that it can scale to large networks with many forwarding requirements, introduces minimal overhead, and quickly reacts to network and controller failures.

CCS Concepts

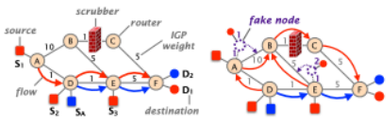
•**Networks** → **Routing protocols;** *Network architectures; Programmable networks; Network management;*

Keywords

Fibbing; SDN; link-state routing

1. INTRODUCTION

Consider a large IP network with hundreds of devices, including the components shown in Fig. 1a. A set of IP addresses (D_1) see a sudden surge of traffic, from multiple entry points (A, D, and E), that congests a part of the network. As a network operator, you suspect a denial-of-service attack (DoS), but cannot know for sure without inspecting the traffic as it could also be a flash crowd. Your goal is therefore to: (i) isolate the flows destined to these IP addresses, (ii) direct them to a scrubber connected between B and C, in order to "clean" them if needed, and (iii) reduce congestion by load-balancing the traffic on unused links, like (B, E).



(a) Initial topology (b) Augmented topology

Figure 1: Fibbing can steer the initial forwarding paths (see (a)) for D_1 through a scrubber by adding fake nodes and links (see (b)).

Performing this routine task is very difficult in traditional networks. First, since the middlebox and the destinations are not adjacent to each other, the configuration of multiple devices needs to change. Also, since intra-domain routing is typically based on shortest path algorithms, modifying the routing configuration is likely to impact many other flows not involved in the attack. In Fig. 1a, any attempt to reroute flows to D_1 would also reroute flows to D_2 since they home to the same router. Advertising D_1 from the middlebox would attract the right traffic, but would not necessarily alleviate the congestion, because *all* D_1 traffic would traverse (and congest) path (A, D, E, B), leaving (A, B) unused. Well-known Traffic-Engineering (TE) protocols (*e.g.*, MPLS RSVP-TE [1]) could help. Unfortunately, since D_1 traffic enters the network from multiple points, many tunnels (three, on A, D, and E, in our tiny example) would need to be configured and signaled. This increases both control-plane and data-plane overhead.

Software Defined Networking (SDN) could easily solve the problem as it enables centralized and direct control of the forwarding behavior. However, moving away from distributed routing protocols comes at a cost. In-

*S. Vissicchio is a postdoctoral researcher of F.R.S.-FNRS.

SIGCOMM '15, August 17 - 21, 2015, London, United Kingdom
 ISBN 978-1-4503-3542-3/15/08...$15.00
DOI: http://dx.doi.org/10.1145/2785956.2787497

How do we create an environment where we foster such creative moments?

My vision on research

(don't quote me on the title)

Dieter Rams (1932–)

Christoph Niemann (1970–)