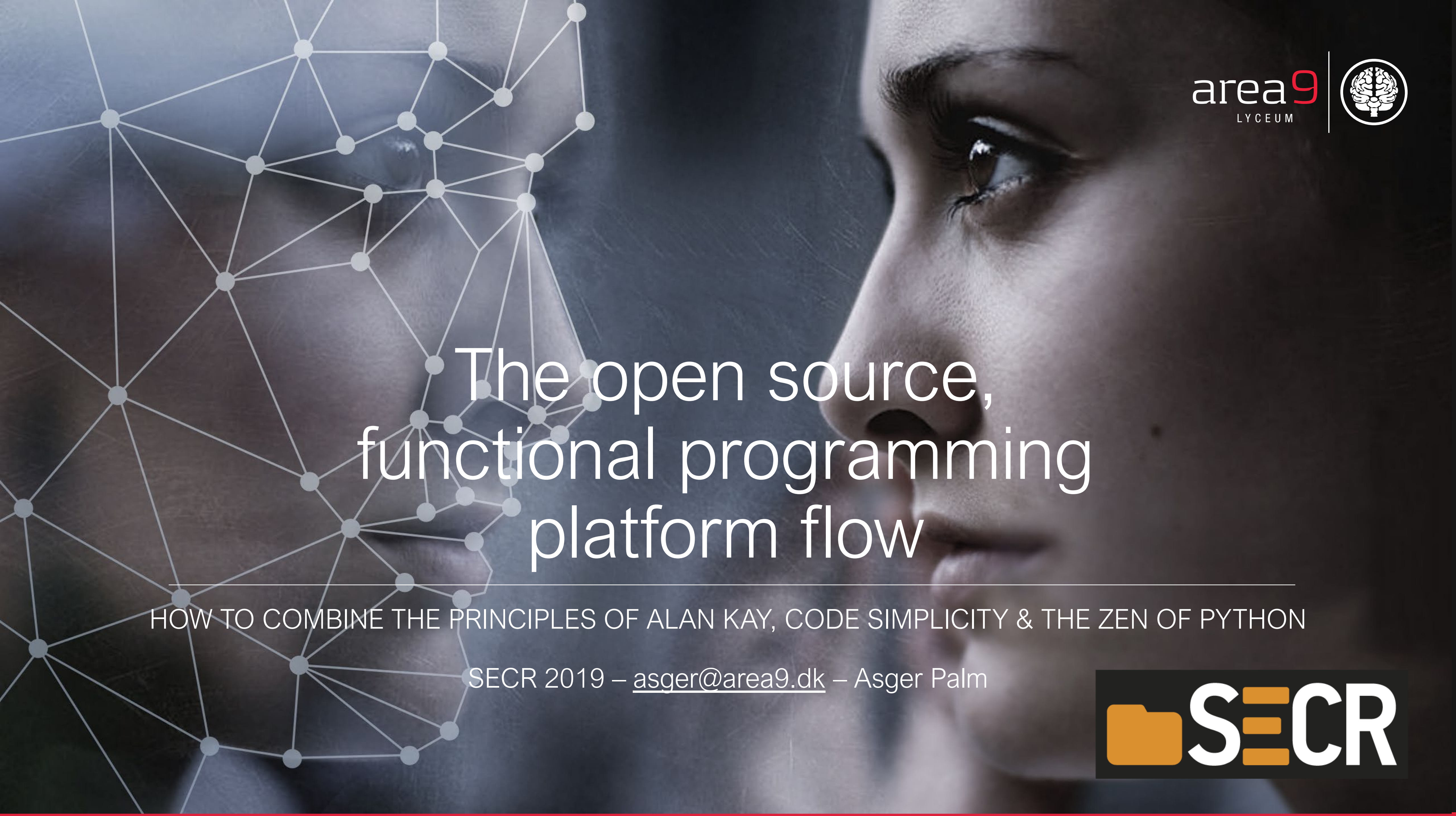


The open source, functional programming platform flow

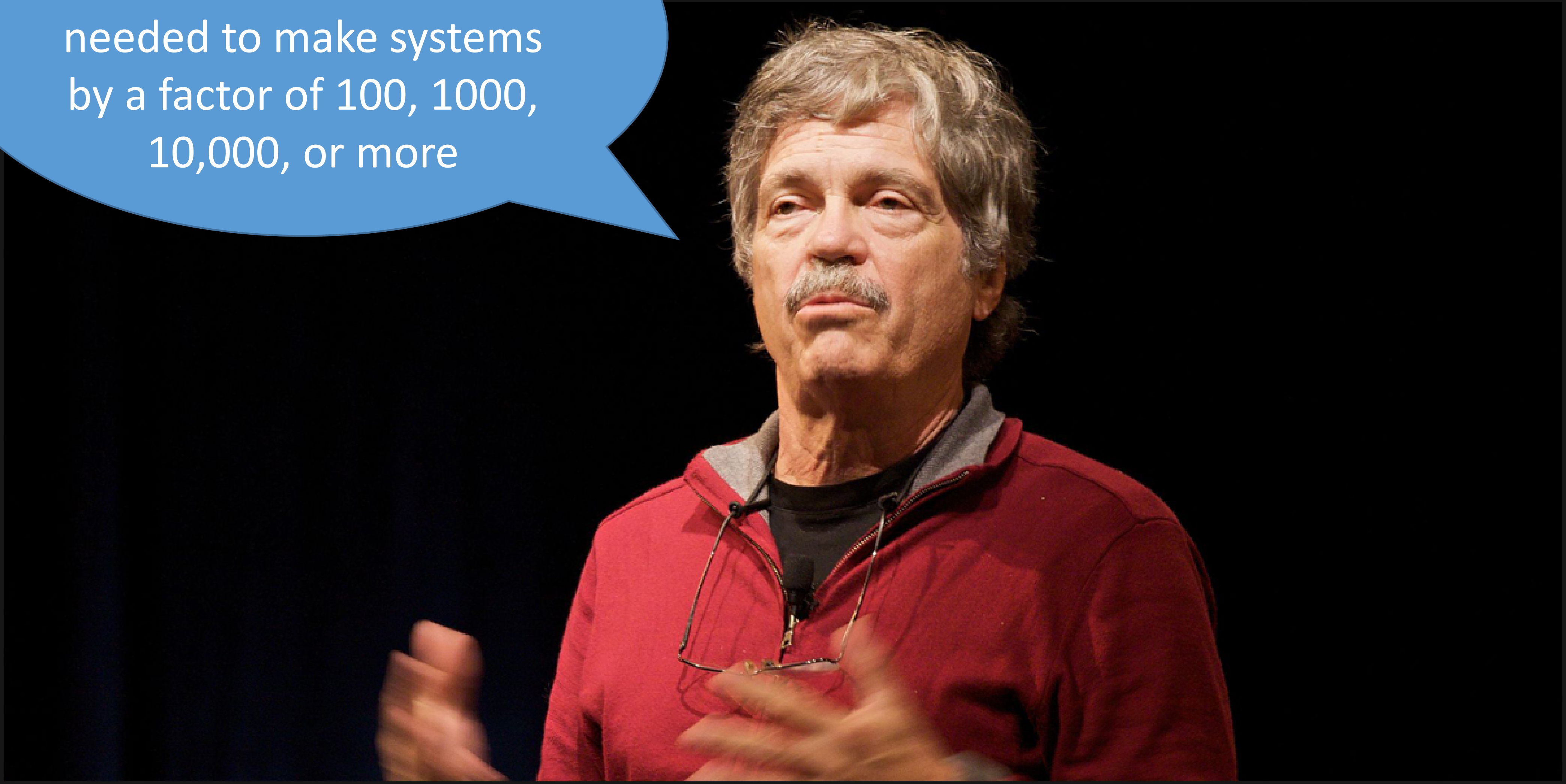
HOW TO COMBINE THE PRINCIPLES OF ALAN KAY, CODE SIMPLICITY & THE ZEN OF PYTHON

SECR 2019 – asger@area9.dk – Asger Palm

area9 LYCEUM

Alan Kay's **STEPS Toward the Reinvention of Programming**

…the aim is to reduce the amount of code needed to make systems by a factor of 100, 1000, 10,000, or more



area9
LYCEUM

Max Kanat-Alexander's **Code Simplicity**

$$\text{desirability} = \frac{\text{future value}}{\text{effort of maintenance}}$$

area9
LYCEUM

Guido van Rossum – **The Zen of Python**

There should be one – and preferably only one – obvious way to do it

The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Reinventing Programming

General Approach—Look for *needs* and identify *kernels* that are close to *needs*, then *create languages* to implement the kernels. Finding kernels is heavily design intensive. Creating languages cleanly and relatively easily is one of the central *needs*, and so several of our kernels are aimed at easily creating new languages. "Powerful principle" heuristics are used to aid the design process. E.g. (a) "Math Wins!", (b) build throwaways to "find the arches", (c) "particles and fields", and (d) "Simulating *time*". And so forth.

"T-Shirt Programming"—One part of the "Math Wins!" principle is drawn from the aesthetic delight of Maxwell's equations on a t-shirt—the basic idea that it is often possible to find/invent mathematics to represent a model which brings all the important relationships into one cosmic eyeful, t-shirt size. This brings forth fanciful and motivational questions about domain areas, such as "How many t-shirts does it (should it) take to model this situation?" In the STEPS project, we ask that question about the whole enterprise, and about the parts of the system, for example "How many t-shirts will be required to define TCP/IP (about 3), or all the graphical operations needed for personal computing? (about 10)".

Code simplicity

The Equation of Software Design: $D=\frac{V}{E}$

where:

D = Desirability of the change.
V = Value of the change.
E = Effort involved in performing the change.

E_i = Effort of implementation
E_f = Effort of maintenance
V_n = Value now
V_f = Value in the future

The Full Equation of Software Design: $D=\frac{V_n+V_f}{E_i+E_m}$

The desirability of a change is directly proportional to the value now plus the future value, and inversely proportional to the effort of implementation plus the effort of maintenance.

Flow

= The Reinvention of Programming

+ Code Simplicity

+ The Zen of Python

Flow = Small + Simple + Powerful

Allows a big team to work together to build maintainable systems with much fewer lines of code compared to other languages

Flow = Standard ML + Curly-brace Syntax + Subtyping

Functional language with static typing
and familiar syntax

Flow = Compiler/JIT + Library + Debugger + Profiler

VS Code/Sublime/Emacs plugins provides productivity

Flow = HTML + iOS + Android + Desktop + Server (Java)

Flow = Material + FRP + FForm + Natives + ...

Flow $= \sum$ Domain Specific Languages

Embedded and external DSLs increase expressivity and compositionality

AREA9 RHAPSODE™

The World's First Four-Dimensional Adaptive Learning Platform area9 LYCEUM

Flow =

Reinvention of Prog.	Code Simplicity	Zen of Python		
Small	Simple	Powerful		
Standard ML	Curly-brace syntax	Subtyping		
Compiler	JIT	Library	Debugger	Profiler
HTML	iOS	Android	Desktop	Server (Java)
Material	FRP	FForm	Natives	
DSL 1	DSL 2	DSL 3	DSL 4	…

↔

http://flow9.org/