

The University of Texas at Dallas School of Arts, Technology, and Emerging Communications (ATEC)  
Richardson, Texas 15 November 2019

# **The Third Era of Design: Systems—Data, Code, + Conversation**

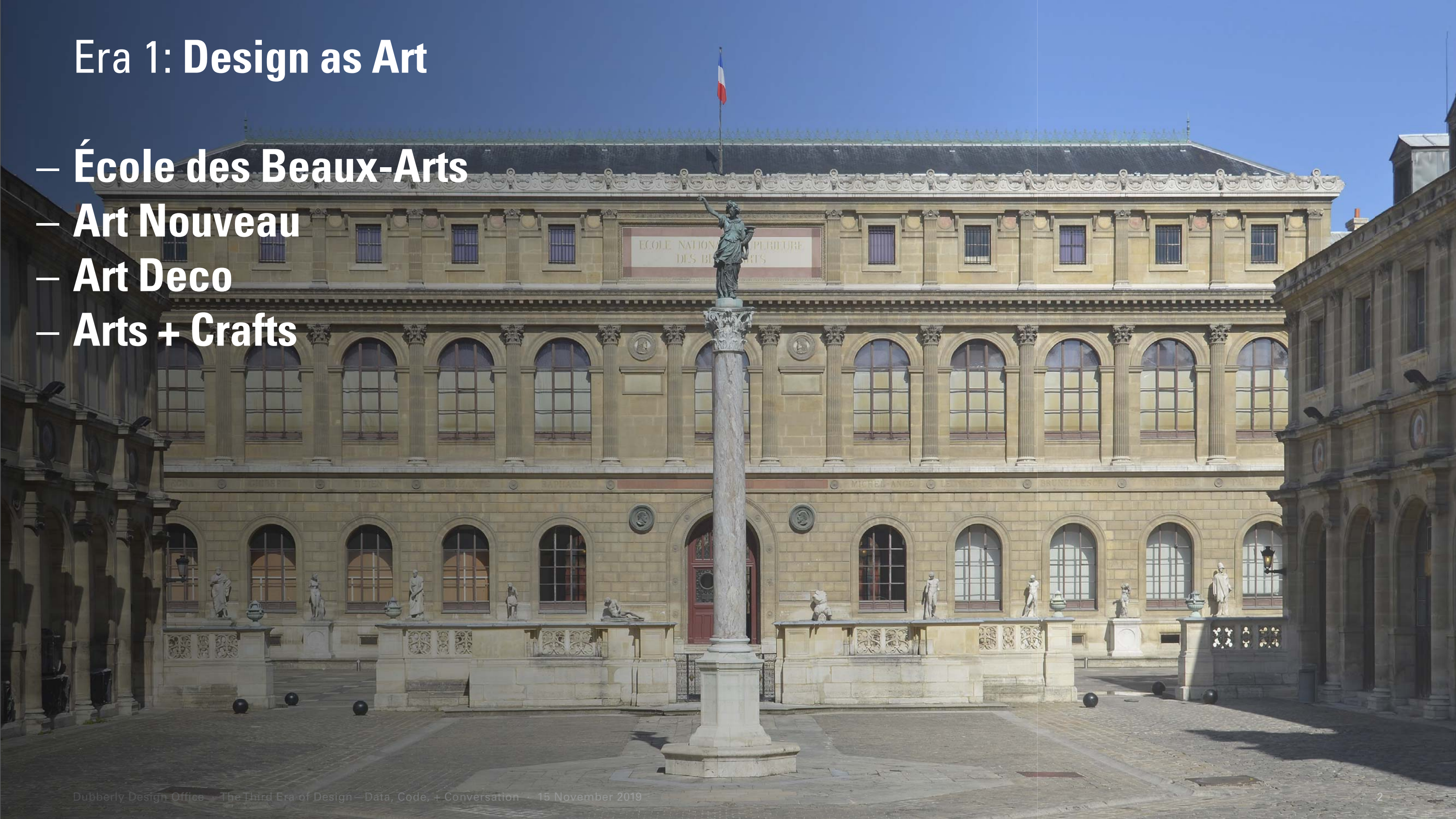
Hugh Dubberly  
Dubberly Design Office

Presentation posted at  
<http://presentations.dubberly.com/thirdera.pdf>



# Era 1: Design as Art

- École des Beaux-Arts
- Art Nouveau
- Art Deco
- Arts + Crafts





# Era 2: Design as Science

- Futurism, *Neue Sachlichkeit*
- Deutscher Werkbund
- Bauhaus, Vkhutemas, HfG Ulm
- Problem Solving / Way Finding
- Design Methods / Design Thinking
- Design Research / Human-centered Design





# Era 3: Design as Systems

- Smart-connected Products
- Platforms, Stacks
- Product-service Ecologies
- Computational Design





**But Google is not a school.  
(Nor is Amazon or Facebook.)**

**And most design schools are rooted in the last century.**

**So: Where might be the future of design?**



# Why not here?





# Origins of Design as Science



# Early modernism began to pose as “objective”.

*“The new art is founded not on a subjective, but on an objective basis. This, like science, can be described with precision and is by nature constructive. It unites not only pure art, but all those who stand at the frontier of the new culture. The artist is companion to the scholar, the engineer, and the worker.”*

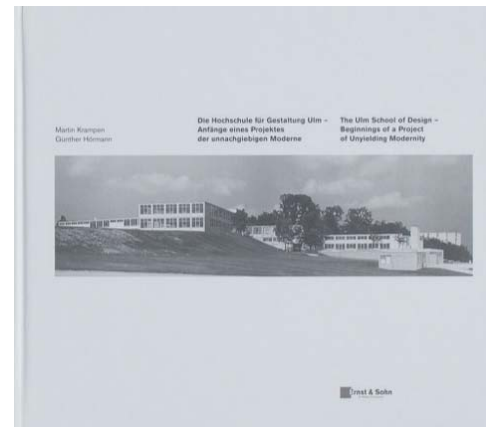


— **El Lissitzky and Illya Ehrenberg**, Statement by the editors of the journal *Veshch*, 1922



# The turn to “objectivity” reached a zenith at HfG Ulm.

*“In all of us [at HfG Ulm], especially myself, there was a deep dissatisfaction with a didactics (and a design activity) that had appealed only to intuition. In this context an increasing interest in disciplines ... with a heuristic function such as ‘problem-solving’ and ‘decision-making’ [showed up]. We were very curious about anything moving in the world that was concerned with scientific questions.”*



— **Tomás Maldonado**, “Looking Back and Forward: Interview,” 2002 [241]



**In the 1960s, the frame of “design-as-science” emerged—  
“problem solving”.**

*“Everyone designs  
who devises **courses of action**  
aimed at changing existing situations  
into preferred ones.”*



— **Herbert Simon**, *Sciences of the Artificial*, 1969



# Lab coats symbolized “objective” professionalism.

*“Unimark designers were the clinicians, diagnosing a client’s problems and then solving them.... Design was scientific and not a messy artistic process. The white lab coat transformed us all into a well-organized team of consistent precise professionals without individuality and quirky intuitions, biases and emotions. Lab coats kept us “clean,” like the “clean” design solutions we sought.”*



— Katherine McCoy, 2019



# Massimo Vignelli + Team (In Lab Coats)





# The problem with problems is:

Whose “problem” is it?

Who defines the problem?

Who frames the situation?

## **Auteur Model of Designing**

Doctor – Patient

Master – Apprentice

**vs.**

## **Facilitator Model of Designing**

Recognizing a “symmetry of ignorance”

Conversation about what we value



Margaret Mead interviewing a subject.

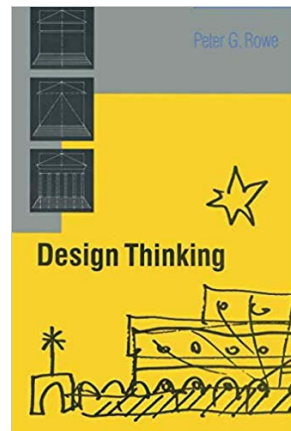


# Not all “problems” are created equal:

**Simple:** Already defined; need solving — also tame, benign  
e.g.,  $2+2=?$ , put a man on the moon

**Complex:** Need definition — also common  
e.g., what should we build?

**Wicked:** Cannot agree on a definition — also mess, tangle  
e.g., poverty, Palestine



— **Peter Rowe**, after Horst Rittel



— **Peter Rowe**, *Design Thinking*, 1987 [39]



**By the 1970s, critics were writing about the social context of design.  
The frame of “design-as-rhetoric” began to appear.**

*“... wicked-problem solving must be understood as an **argumentative process**: one of raising questions and issues towards which you can assume different positions, with evidence gathered and arguments built for and against these different positions.”*



— **Horst Rittel**, “On the Planning Crisis: Systems Analysis of the ‘First and Second Generations,’” 1972



# Origins of Design as Systems

*“... a building cannot be viewed simply in isolation.  
It is only meaningful as a human environment.  
It perpetually interacts with its inhabitants,  
on the one hand serving them  
and on the other hand controlling their behavior.*

*In other words structures make sense  
as parts of larger **systems that include human components**  
and the architect is primarily concerned  
with these larger systems;  
they (not just the bricks and mortar part)  
**are what architects design.”***



**— Gordon Pask, *The Architectural Relevance of Cybernetics*, 1967**



# A matrix of design: the six types

Jay Doblin, 1987

Tangible objects and messages

---

**Appearance Products**

Christmas ornaments  
Medals  
Trophies

---

**Performance Products**

Crowbars  
Paper clips

Sets of coordinated products  
and the people who operate them

---

**Appearance Unisystems**

Restaurant environment  
South Street Seaport  
Disneyland

---

**Performance Unisystems**

Compact kitchen  
NASA space mission  
United Airlines

Competing unisystems

---

**Appearance Multisystems**

The fashion industry

---

**Performance Multisystems**

The airline industry  
The computer industry

From “A Short, Grandiose Theory of Design,” STA Design Journal

# Era analysis: evolution of design

Joi Ito, 2017

---

**Objects** (physical and immaterial)

---

**Systems**

---

**Complex Adaptive Systems**

*“Design has also evolved  
from the design of objects  
both physical and immaterial,  
to the design of systems,  
to the design of complex adaptive systems.*

*This evolution is shifting the role of designers;  
they are no longer the central planner,  
but rather participants  
within the systems they exist in.  
This is a fundamental shift —  
one that requires a new set of values.”*

— **Joi Ito**, “Design and Science,” January 11, 2016



# John Maeda has offered a sort of era analysis.

## 1 Classical Design

There is a right way to make what is perfect, crafted, and complete.

## 2 Design Thinking

Because execution has outpaced innovation, and experience matters.

## 3 Computational Design

Design for billions of individual people and in real time, is at scale and TBD.

—Design in Tech Report, 2018

**Stephen Anderson says, “The future of design is complexity + computation.”**

**Design 1.0**  
**Product**

**Design 2.0**  
**Experience**

**Design 3.0**  
**Outcomes**

—<https://medium.com/@stephenanderson/the-future-of-design-computation-complexity-a434d2da3cd5>

# Richard Buchanan proposed “four orders of design.”

- 1 **Communications** —  
a focus on meaning and symbols
- 2 **Artifacts** —  
a focus on form and things
- 3 **Interactions** —  
a focus on behavior and action
- 4 **Fourth order** —  
a focus on “environments and systems in which all other orders exist”



# Levels of Systems

the level of <b>Frameworks</b>	Only the geography and anatomy of the subject is described and analyzed; a kind of system of static relations. [Most architecture and graphic design systems are of this type.]
the level of <b>Clockworks</b>	Machines that are determined.
the level of <b>Thermostats</b>	The level of control in mechanical and cybernetical [sic] systems.
the level of the <b>Cell</b>	As an open and self-maintaining system, having a throughput that transforms unpredicted inputs into outputs [what Maturana, Varela, and Uribe later called an “autopoetic” system].
the <b>Genetic</b> and <b>Societal</b> level	Of plants and accumulated cells.
the level of the <b>Animal</b>	Specialized receptors, a nervous system, and an “image”.
the <b>Human</b> level	All of the previous six—plus self-consciousness. The system knows that it knows, and knows that it dies.
the level of the <b>Social Organism</b>	The unit at this level is a role, rather than a state; messages with content and meaning exist, and value systems are developed.
the level of <b>Transcendental</b> systems	The “ultimates” and “absolutes” and the “inescapables” with systematic structure.

— Kenneth Boulding, 1956

# What is Computational Design?



# Computing as Designing Programmes, for coherence + flexibility.

Also known as Design Systems  
e.g., Google's "Material Design" (System)

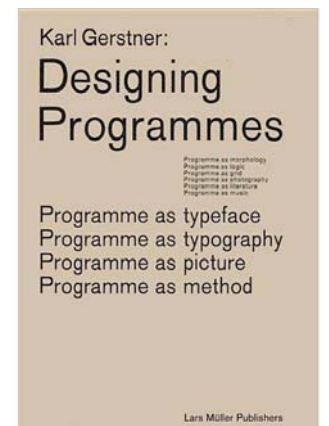
**Theme and variation**

**Elements and rules (for relating the elements)**

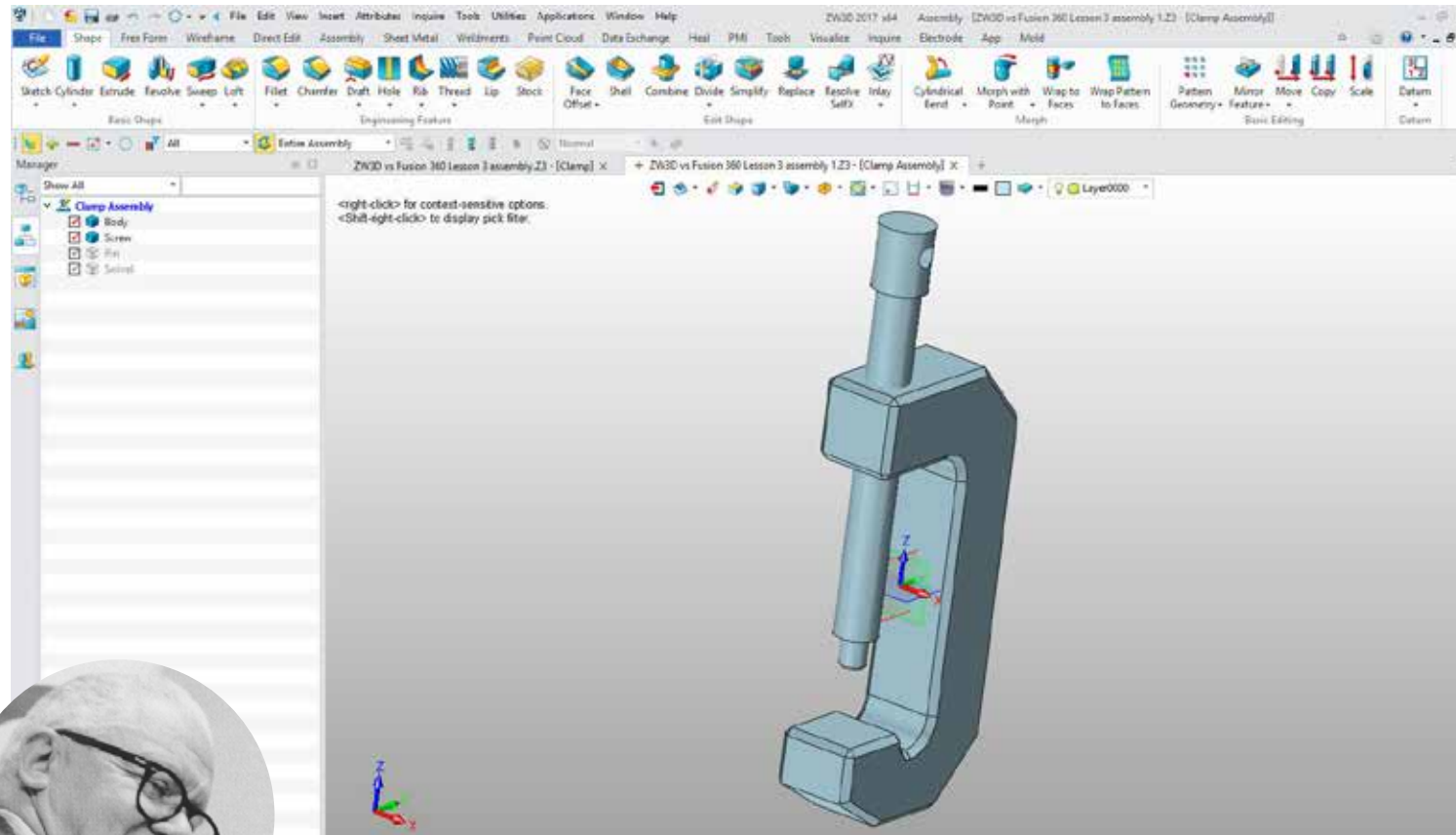
**And meta-rules for adding elements**

**And meta-meta-rules for changing the rules.**

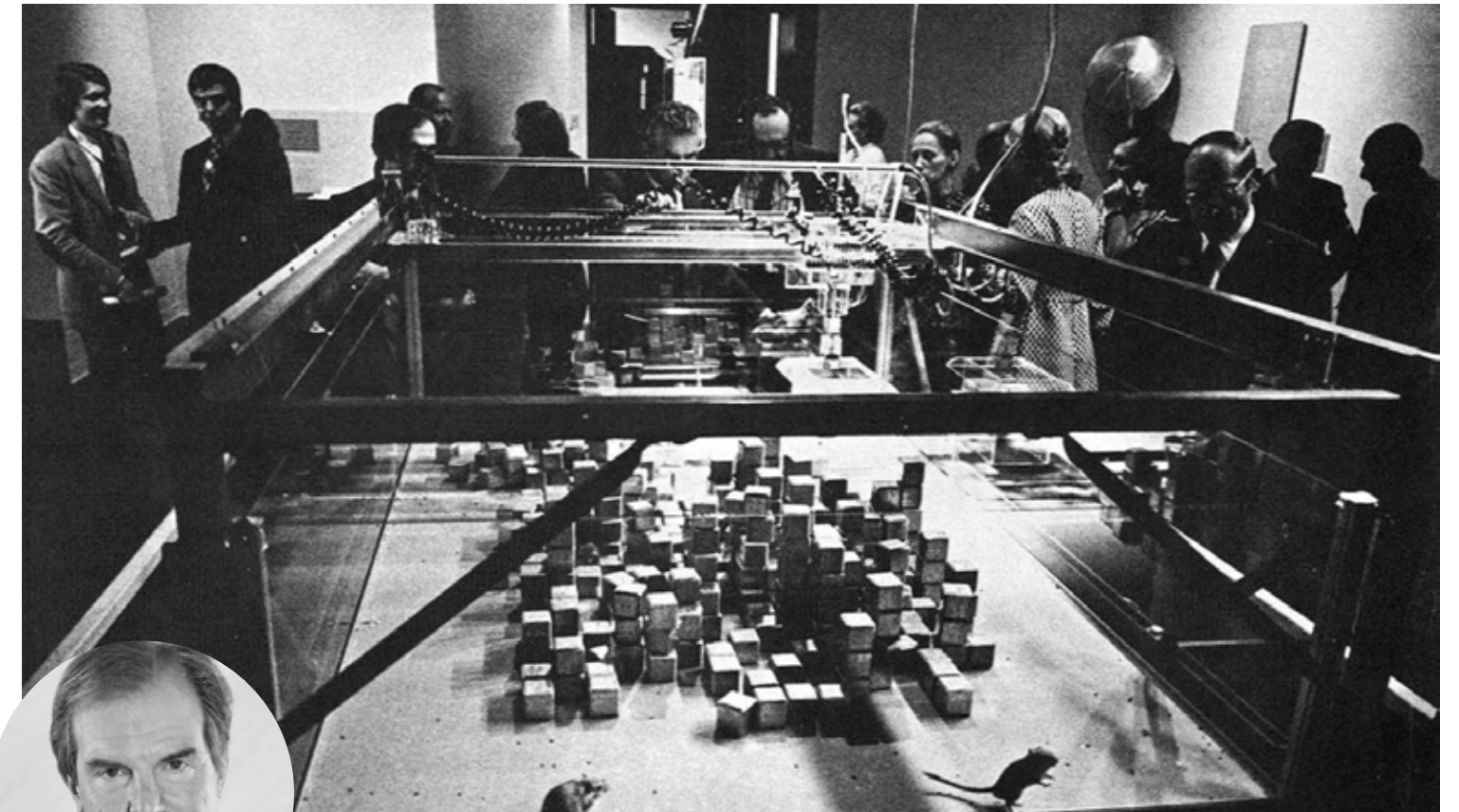
More on Design Systems in a few minutes...



# Computing as Tool, augmenting the design process.



*Paul Rand*

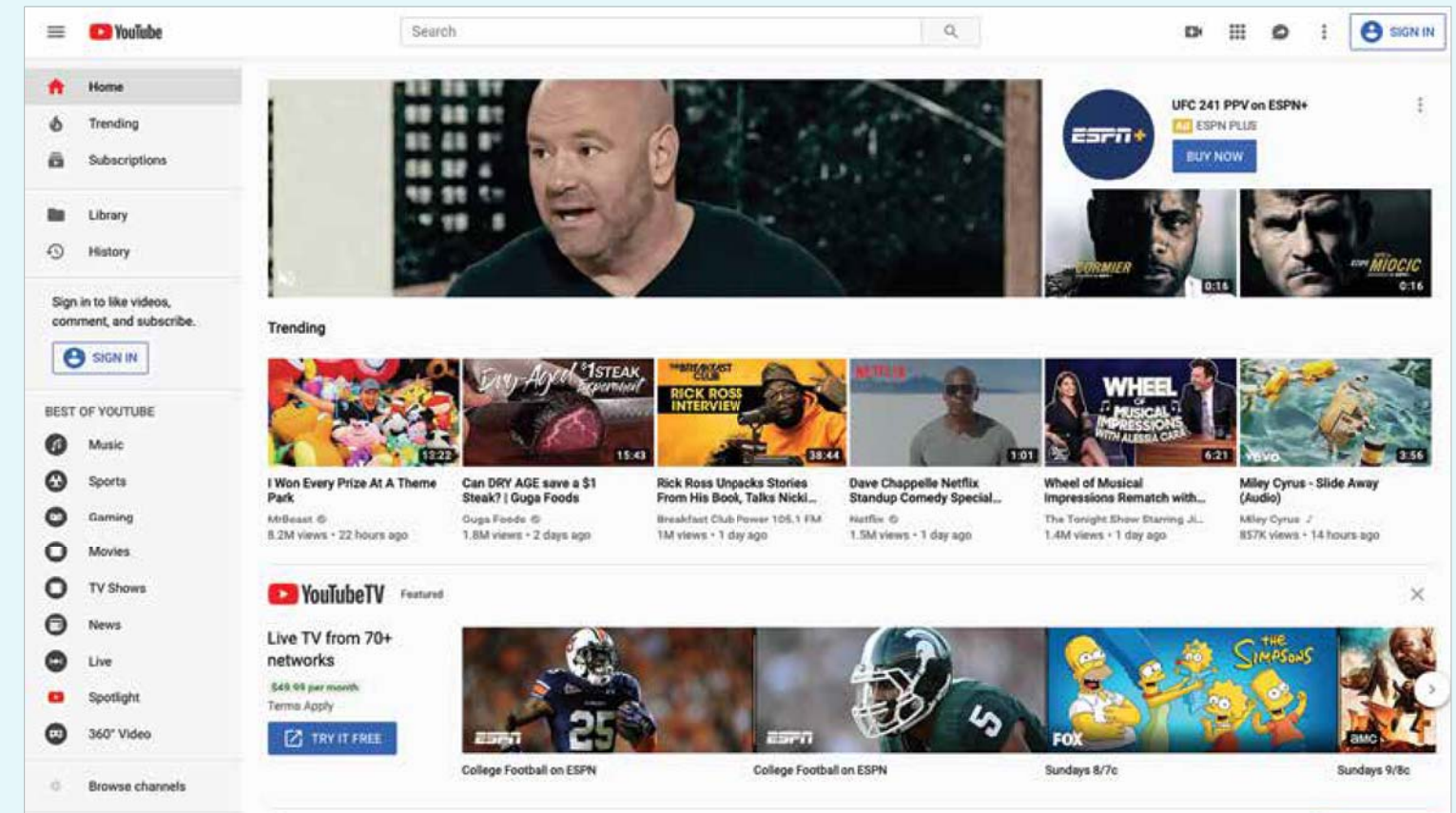


*Nicholas Negroponte*

**From production tool, e.g. AutoCAD**

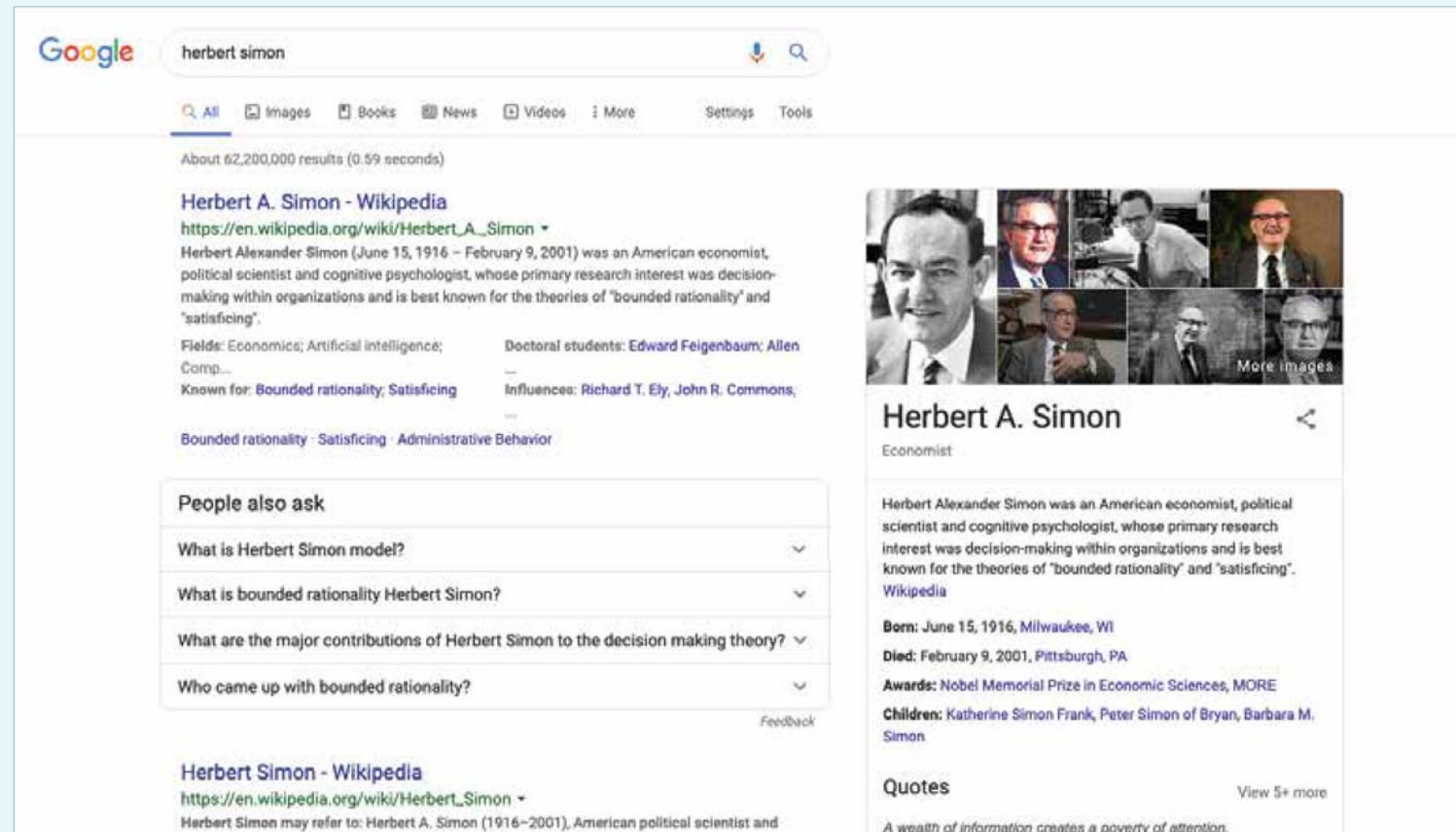
**To collaboration partner, e.g. the Architecture Machine**





## And for entertainment

# Computing as Material, to be shaped into products.

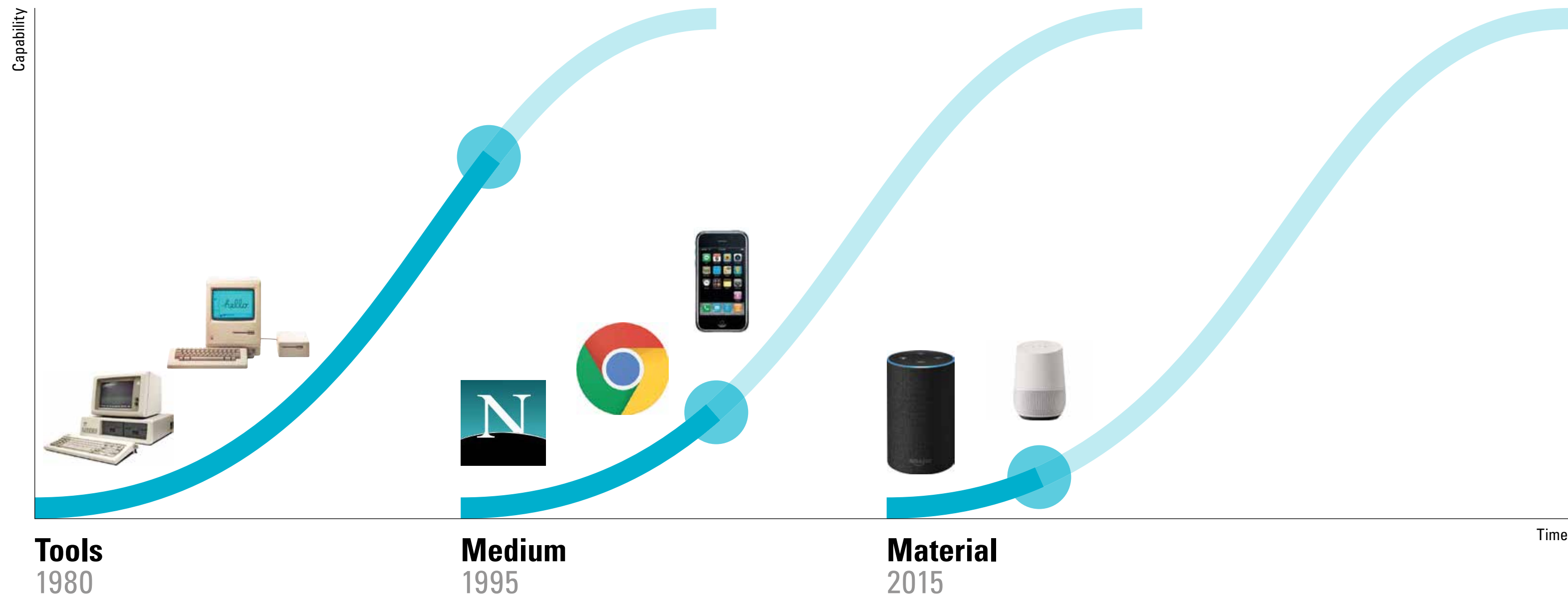


For good

And for evil



# Each of these “digital transformations” is at a different stage.



# What is Computing as Material?

- Predictive Analytics
- Anticipatory Computing
- Design for Conversation
- Solution Space Modeling
- Generative Design

# **Predictive Analytics is pattern-finding software— algorithms making sense of measurements:**

- Classical Methods of Statistics
- Artificial Intelligence (so-called AI)
- Deep Learning (DL)
- Machine Learning (ML)
- Computer Vision (CV)
- Natural Language Processing (NLP)



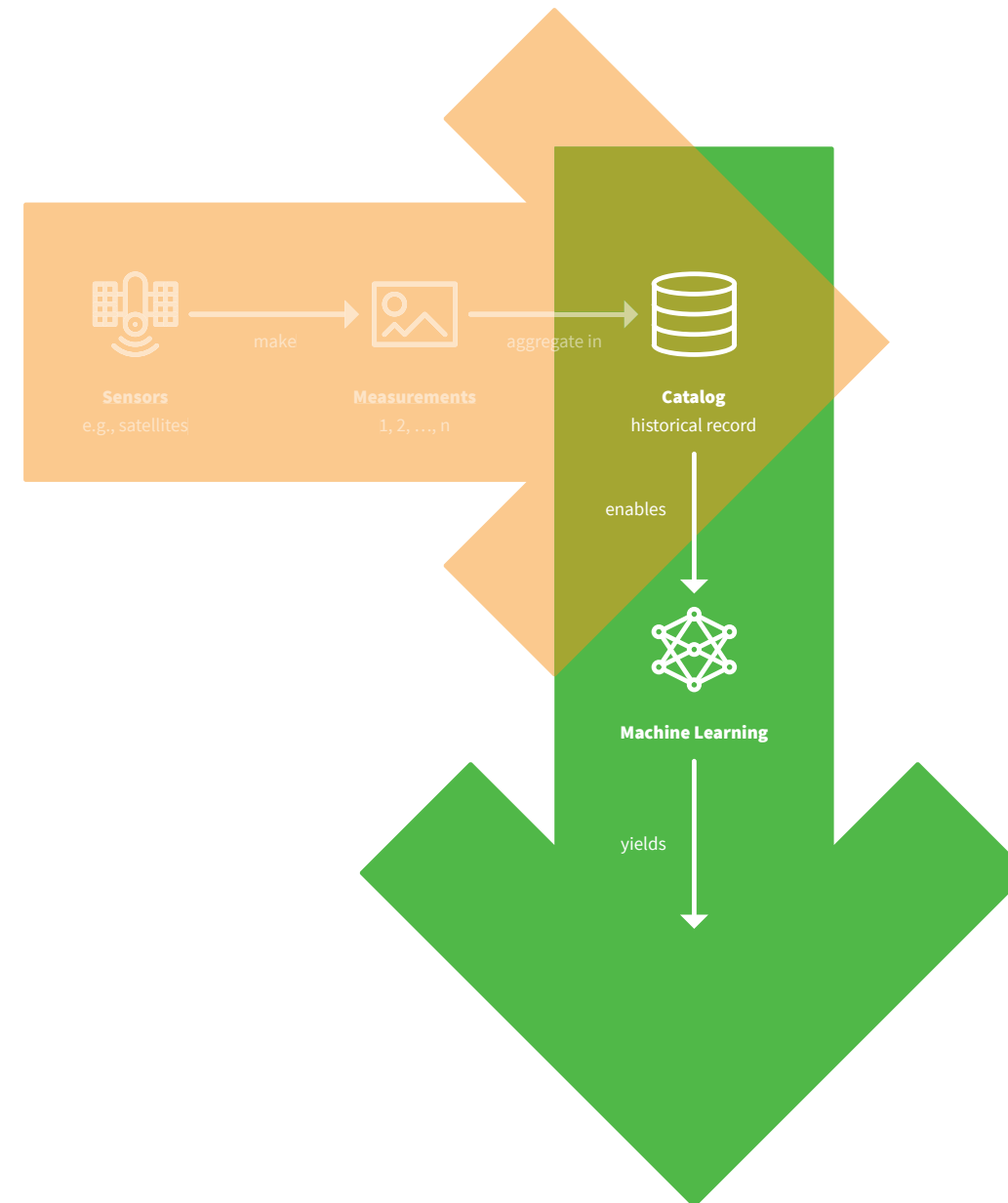
# Predictive Analytics—collecting old data.

## 1. Gather histories

Sensors make a series of point in time measurements. As measurements accumulate, a historical record emerges.



# Predictive Analytics—training the system.



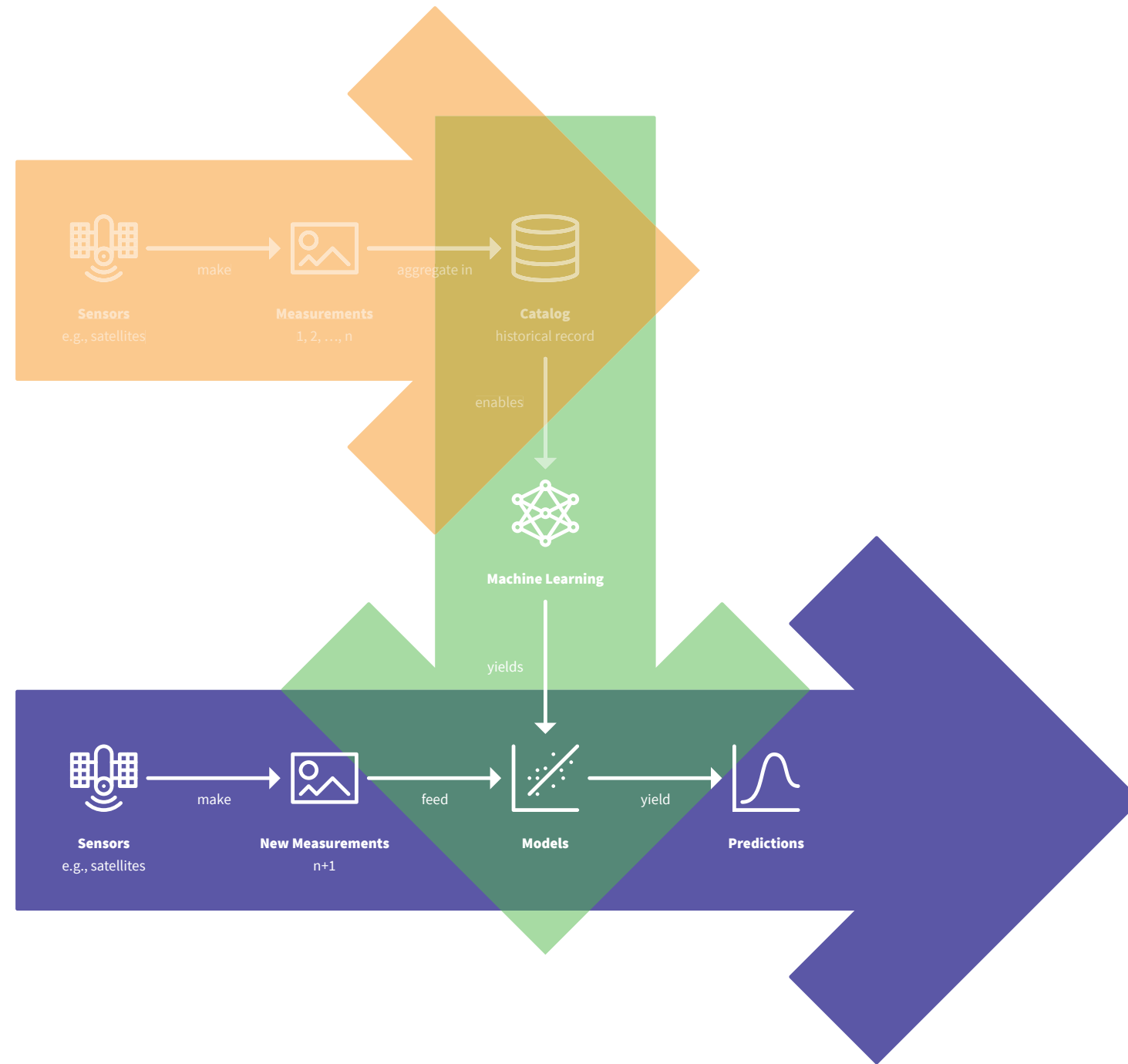
## 2. Derive models

Sufficient historical data enables analysts to discover patterns and relationships—these are codified in models.

# Predictive Analytics—analyzing new data.

## 3. Predict futures

Once trained, new measurements are fed through the model to predict the future—enabling us to act today.

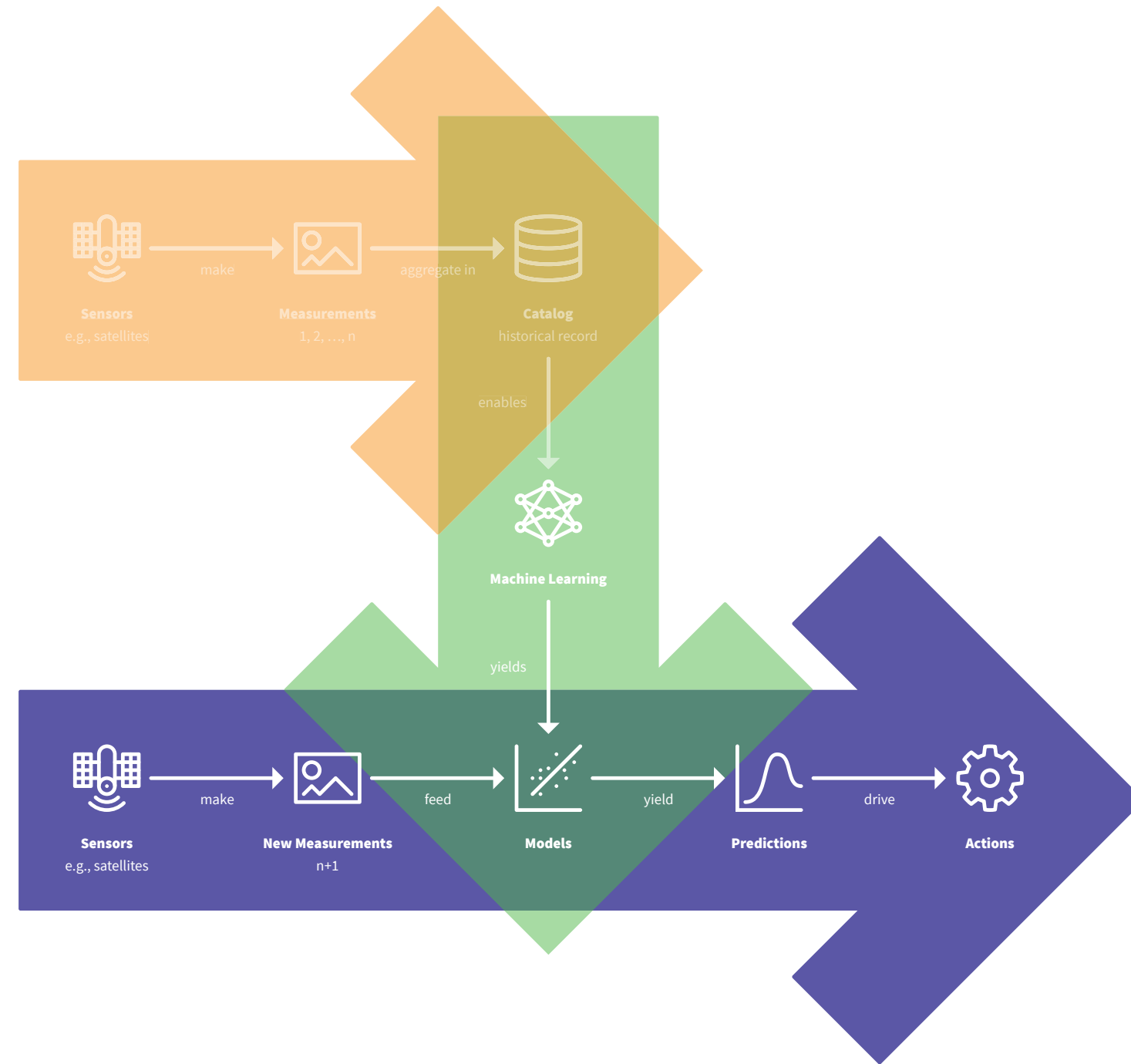




# Anticipatory Computing—acting on the prediction.

## 3. Predict futures

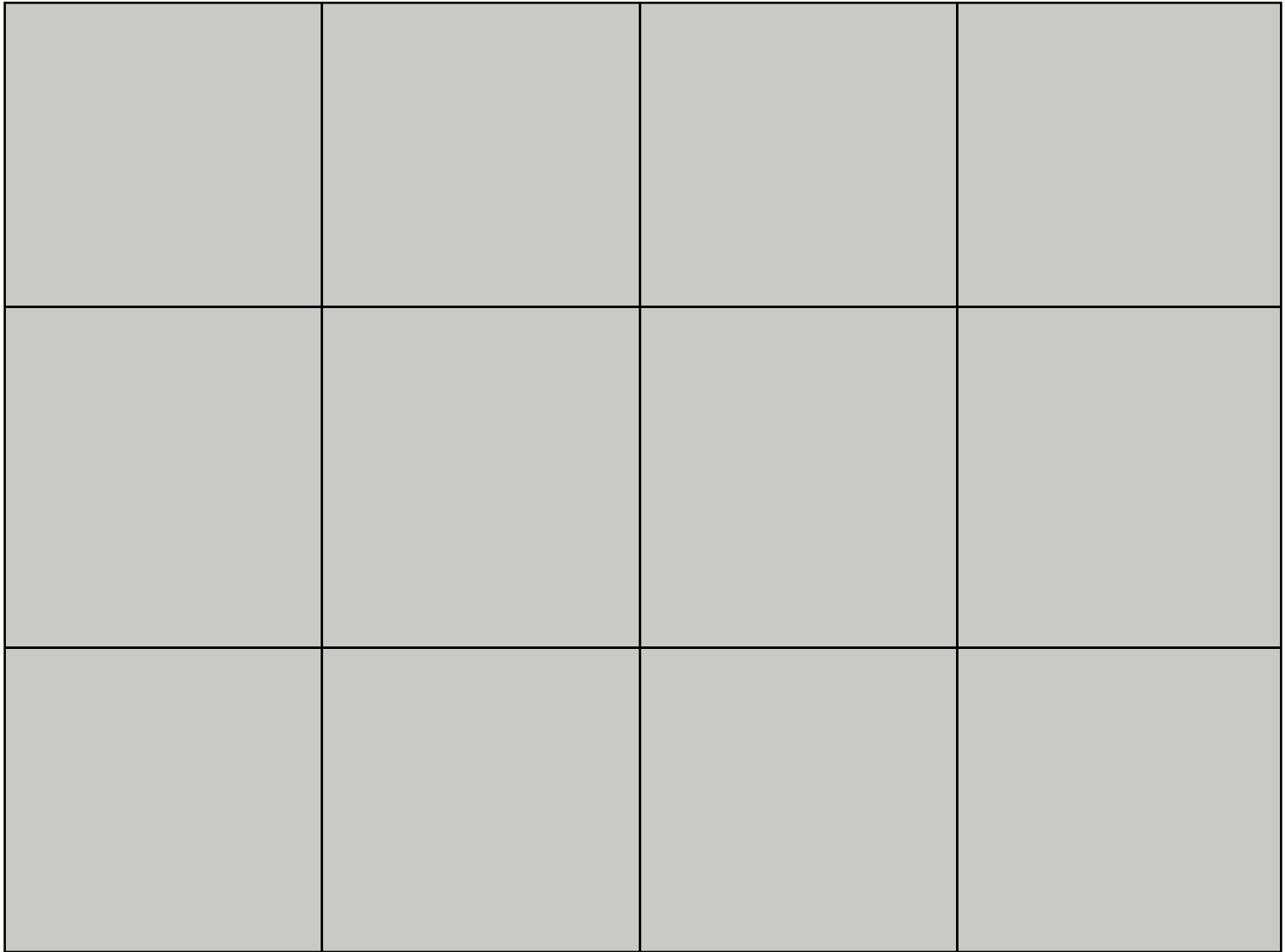
Once trained, new measurements are fed through the model to predict the future—enabling us to act today.



# Design for Conversation—e.g., Negroponte’s Architecture Machine

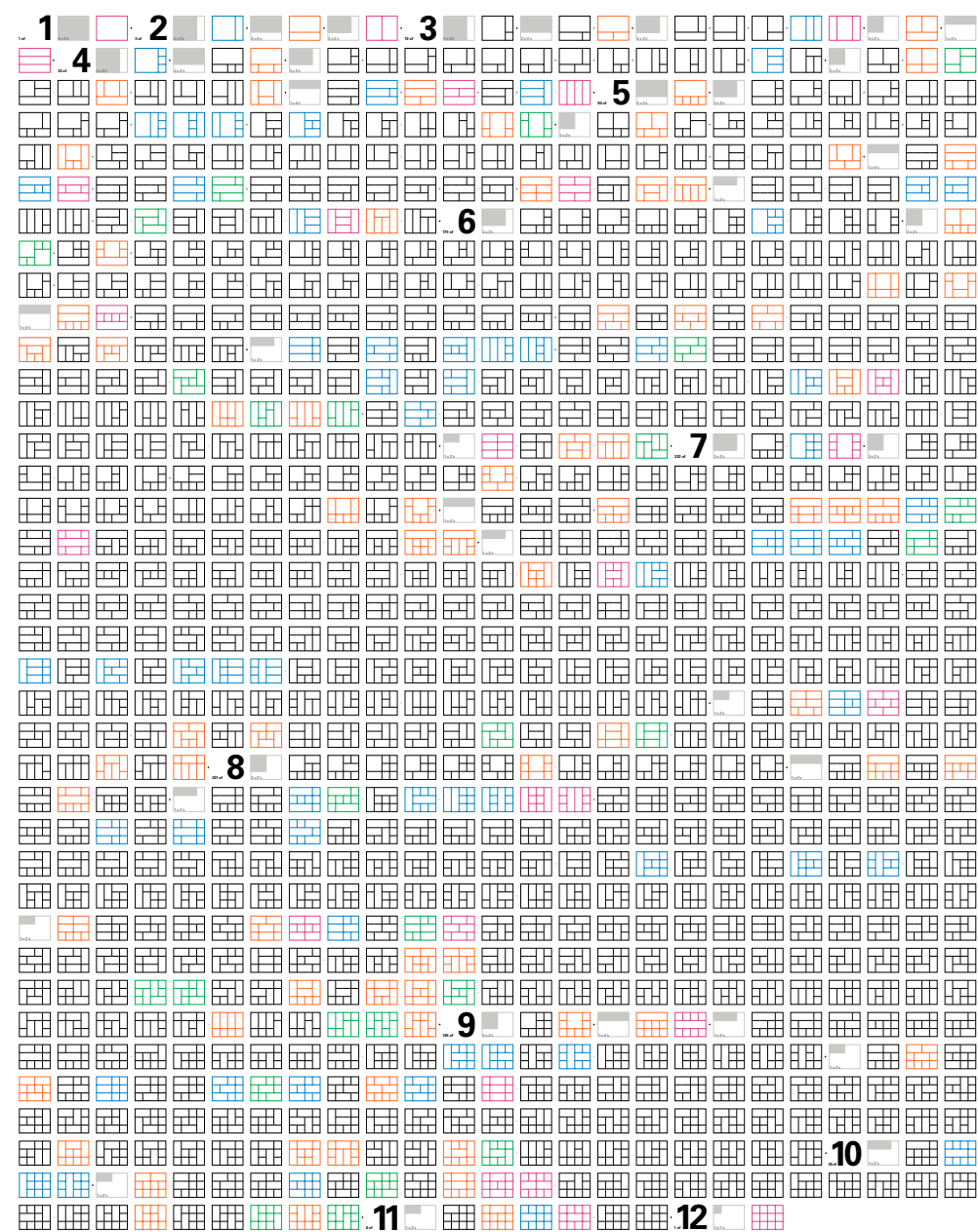


# Solution Space Modeling—e.g., 3x4 grid permutations



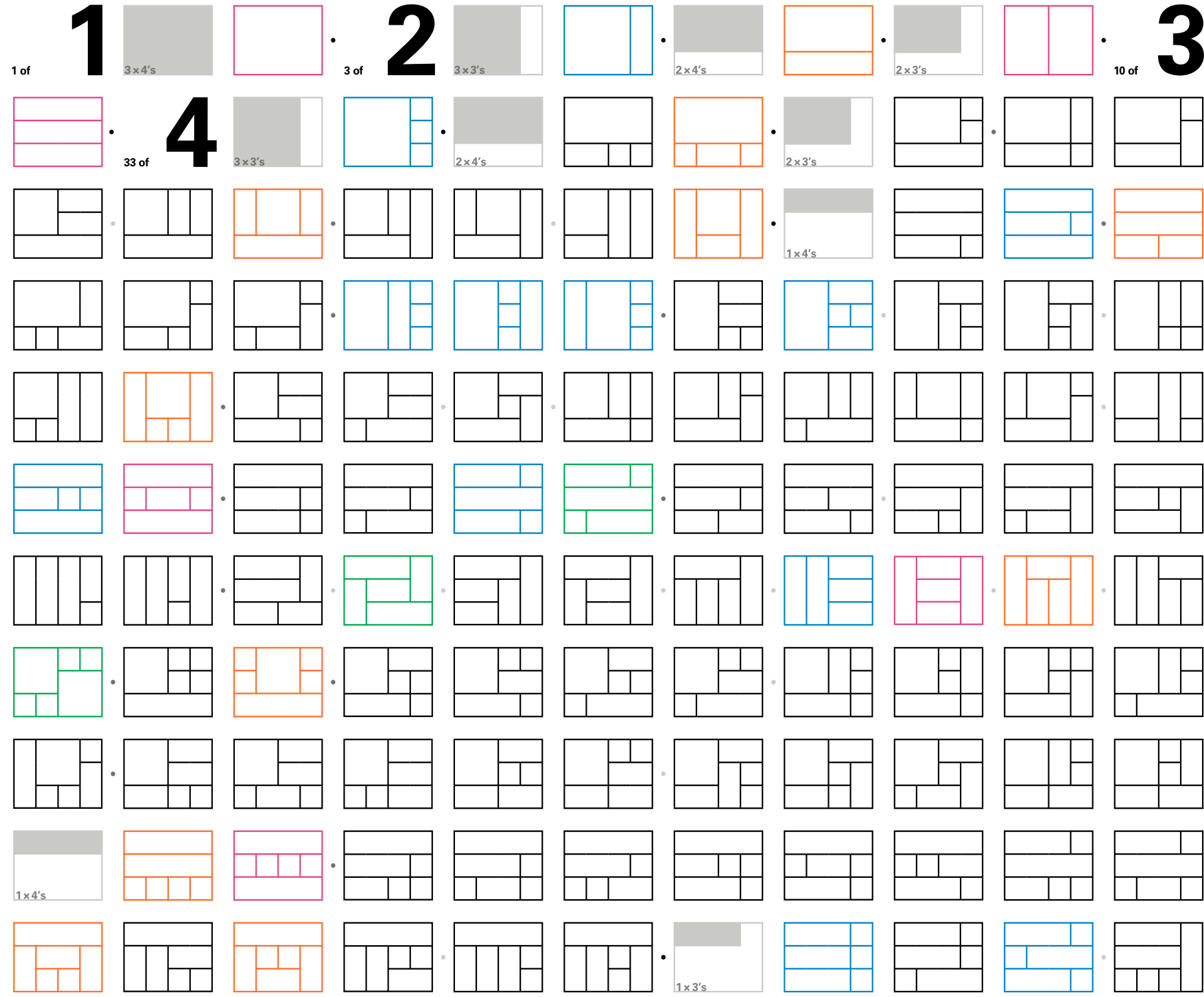


# Solution Space Modeling

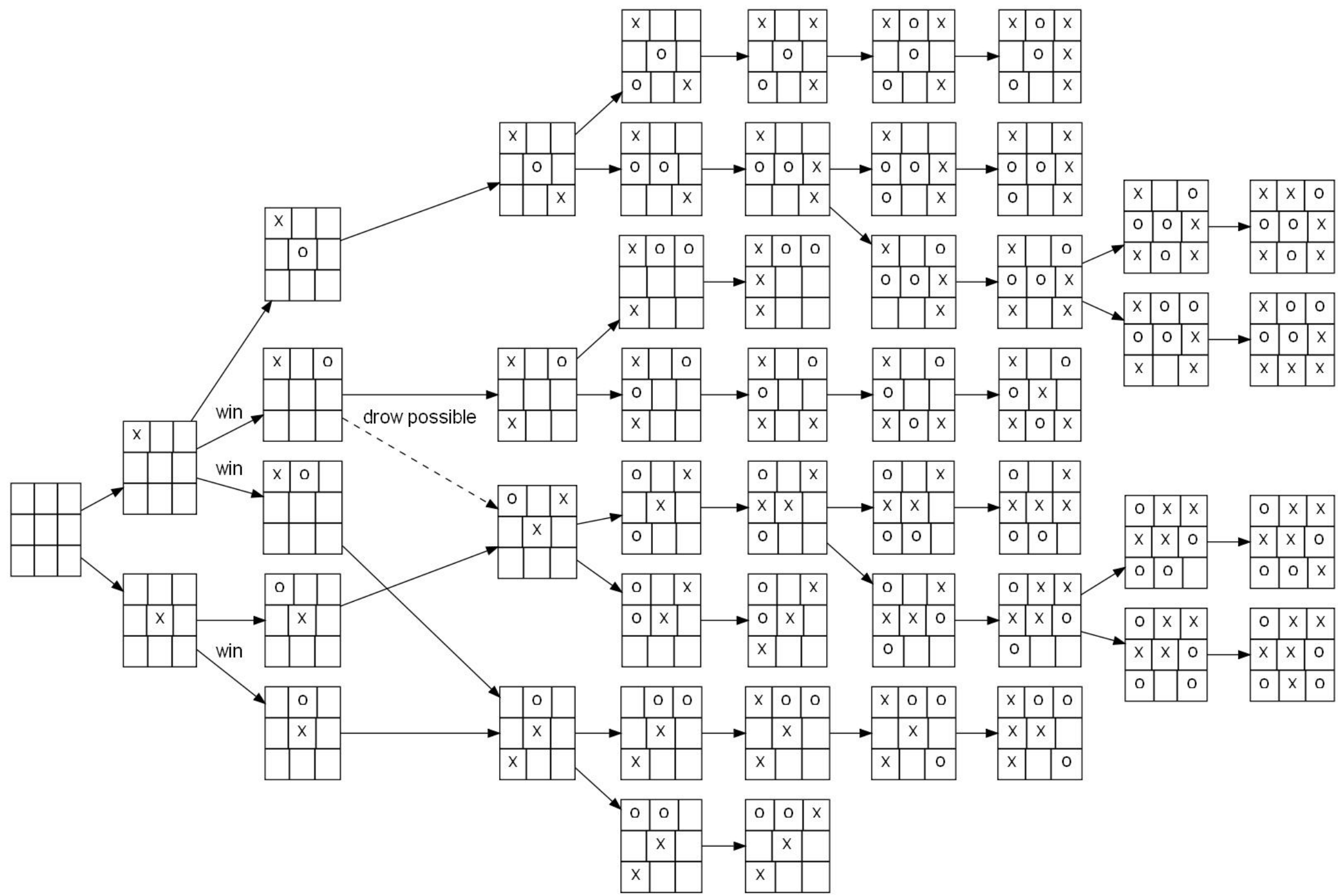


The 892 unique ways to partition a 3x4 grid

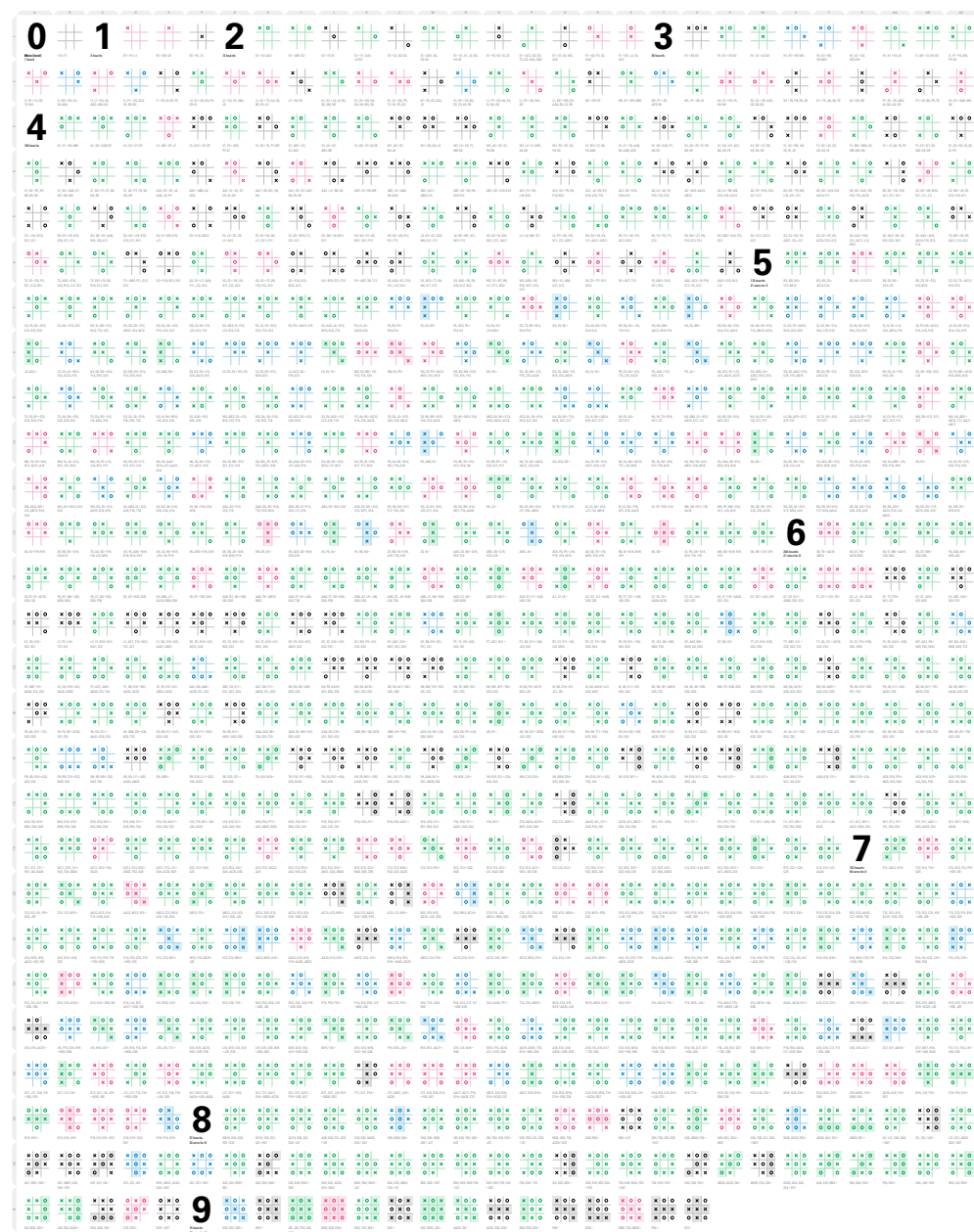
This poster illustrates a design in design problem. Consider the 3x4 grid. It is a rectangle that can be partitioned into smaller rectangles. The problem is to find all possible partitions of the 3x4 grid into smaller rectangles. The solution is to use a recursive algorithm to generate all possible partitions. The algorithm starts with a 3x4 grid and recursively partitions it into smaller rectangles until no more partitions are possible. The result is a set of 892 unique partitions. The partitions are color-coded by the number of 1x4 and 4x1 rectangles they contain. The grid is organized into 12 columns, each representing a different partition type. The partitions are shown in a variety of colors including grey, blue, orange, pink, and green.



# Solution Space Modeling—e.g., Tic-Tac-Toe



# Solution Space Modeling



## The tic-tac-toe solution space

**1.1.1.1** **Introduction**

Algorithms are systematic methods for solving a problem. They are a sequence of instructions that a computer can follow to perform a task. Algorithms are used in many fields, including mathematics, science, engineering, and business.

Designing an algorithm involves several steps:

1. **Problem Definition:** Clearly define the problem you want to solve.
2. **Input and Output:** Determine what data you need to start with and what results you want to produce.
3. **Algorithm Design:** Develop a step-by-step plan to solve the problem.
4. **Implementation:** Write the algorithm in a programming language.
5. **Testing and Debugging:** Run the program with test data to ensure it works correctly.

Algorithms are essential for solving complex problems efficiently. They provide a structured approach to problem-solving and are the foundation of computer science.

**1.1.1.2** **Algorithm Design**

Algorithm design is the process of creating a plan to solve a problem. It involves breaking down a problem into smaller, manageable parts and developing a sequence of steps to solve each part.

Common algorithm design techniques include:

- Divide and Conquer:** Break a problem into smaller sub-problems, solve each sub-problem, and then combine the solutions.
- Dynamic Programming:** Solve a problem by breaking it down into overlapping sub-problems and storing the results of sub-problems to avoid redundant calculations.
- Greedy Algorithms:** Make the locally optimal choice at each step with the hope of finding a global optimum.
- Backtracking:** Build a solution incrementally, removing any steps that are shown to be invalid.
- Brute Force:** Try all possible solutions until the correct one is found.

Algorithm design is a critical skill for computer scientists and engineers. It allows them to solve complex problems efficiently and effectively.

**1.1.1.3** **Algorithm Analysis**

Algorithm analysis is the process of evaluating the performance of an algorithm. It involves determining the time and space complexity of an algorithm.

**Time Complexity:** Measures the amount of time an algorithm takes to run. It is often expressed using Big O notation.

**Space Complexity:** Measures the amount of memory an algorithm uses. It is also often expressed using Big O notation.

Understanding algorithm analysis is crucial for selecting the most efficient algorithm for a given problem. It helps in predicting the performance of an algorithm and in identifying areas for optimization.

**1.1.1.4** **Algorithm Implementation**

Algorithm implementation is the process of converting an algorithm into a program. It involves writing code in a programming language that implements the steps of the algorithm.

Key considerations for algorithm implementation include:

- Language Choice:** Selecting a programming language that is suitable for the task.
- Code Style:** Writing clean, readable, and maintainable code.
- Testing:** Thoroughly testing the program to ensure it works correctly.
- Optimization:** Optimizing the code for better performance.

Algorithm implementation is a critical step in the process of solving a problem. It ensures that the algorithm is correctly translated into a form that can be executed by a computer.

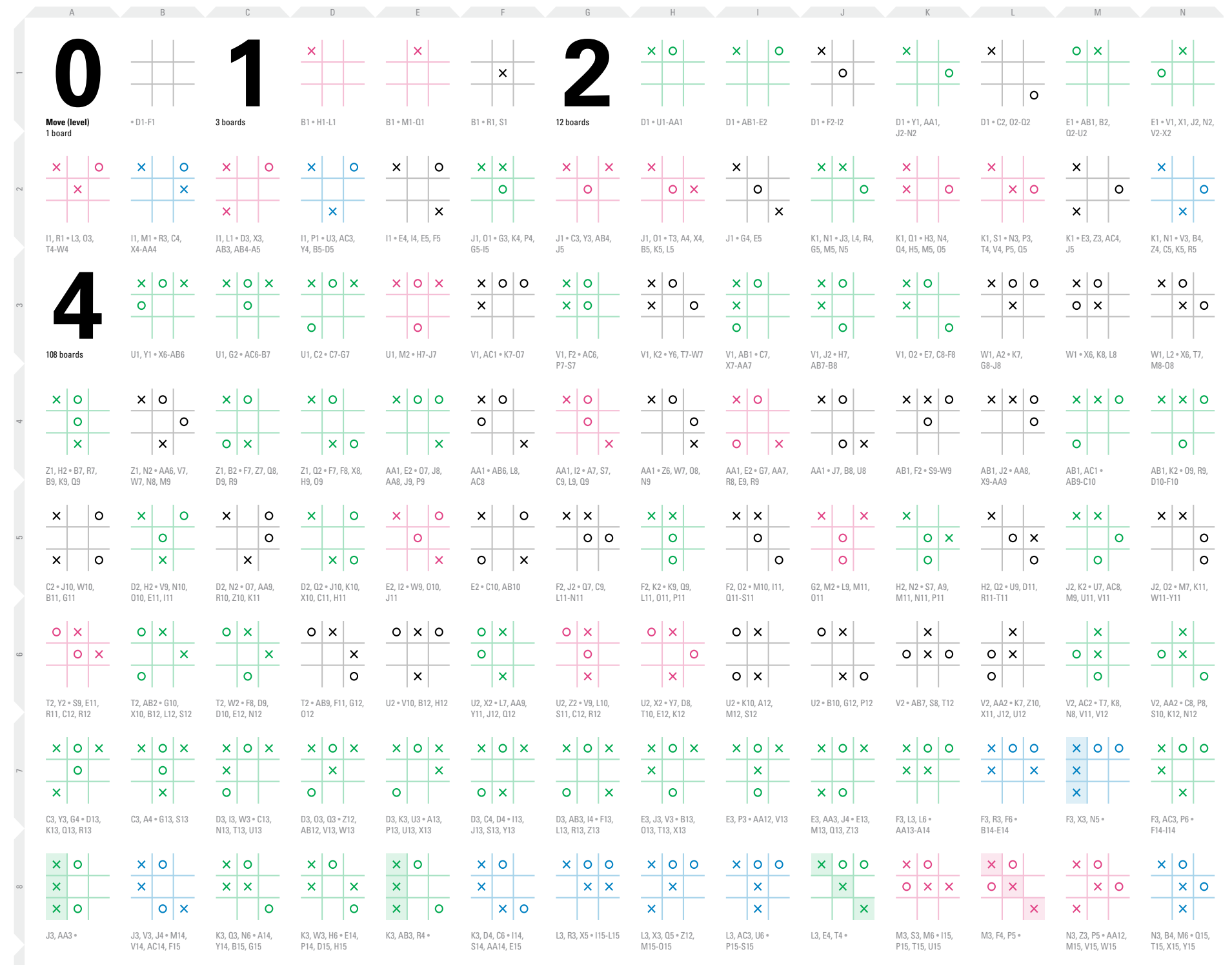
**1.1.1.5** **Algorithm Evaluation**

Algorithm evaluation is the process of assessing the performance of an algorithm. It involves comparing the results of an algorithm against a set of criteria.

Common evaluation criteria include:

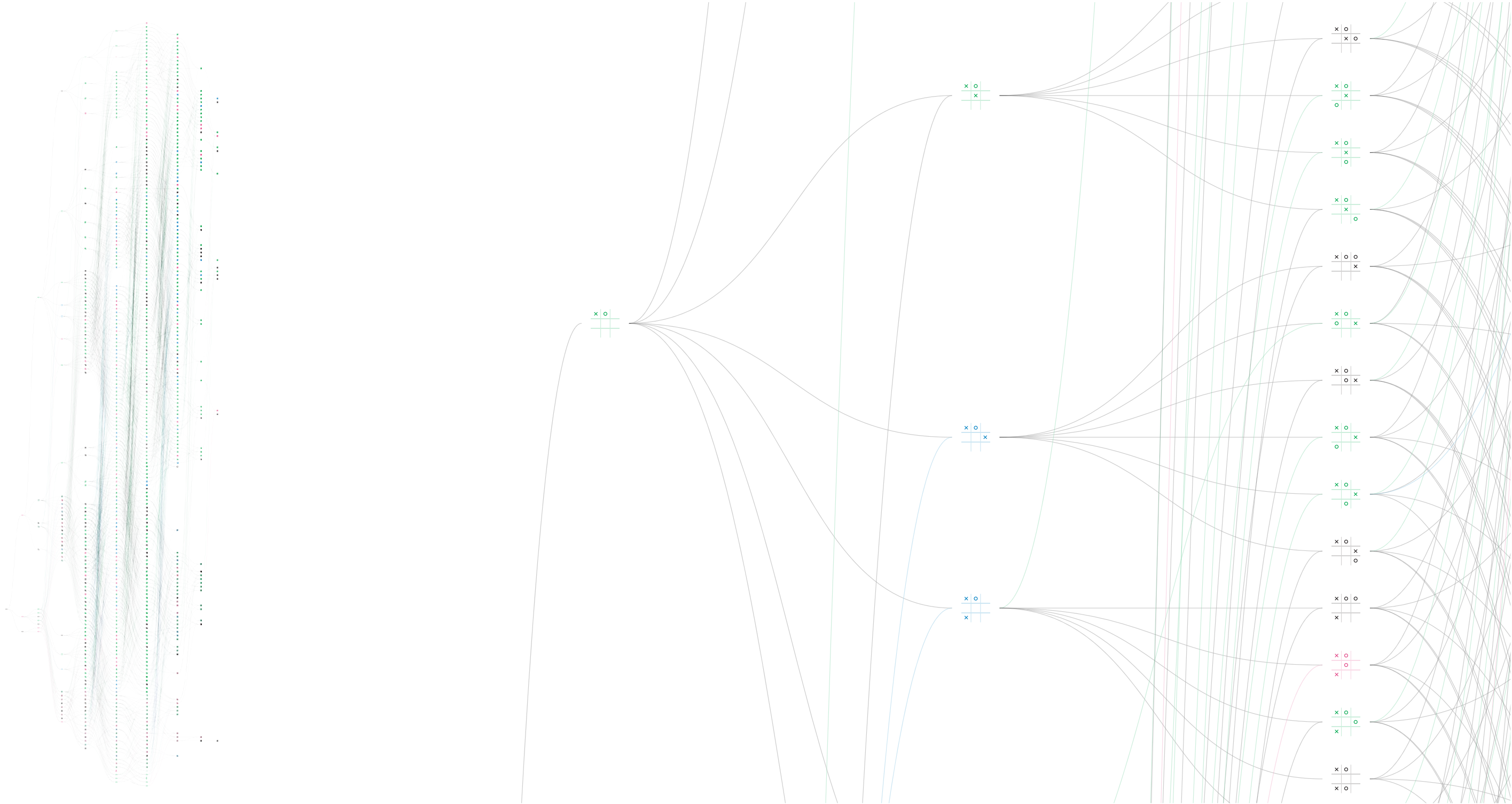
- Accuracy:** How close the results are to the correct answer.
- Efficiency:** How quickly the algorithm runs.
- Scalability:** How well the algorithm performs as the problem size increases.
- Robustness:** How well the algorithm handles unexpected input or errors.

Algorithm evaluation is essential for determining the effectiveness of an algorithm. It helps in identifying the strengths and weaknesses of an algorithm and in making improvements.

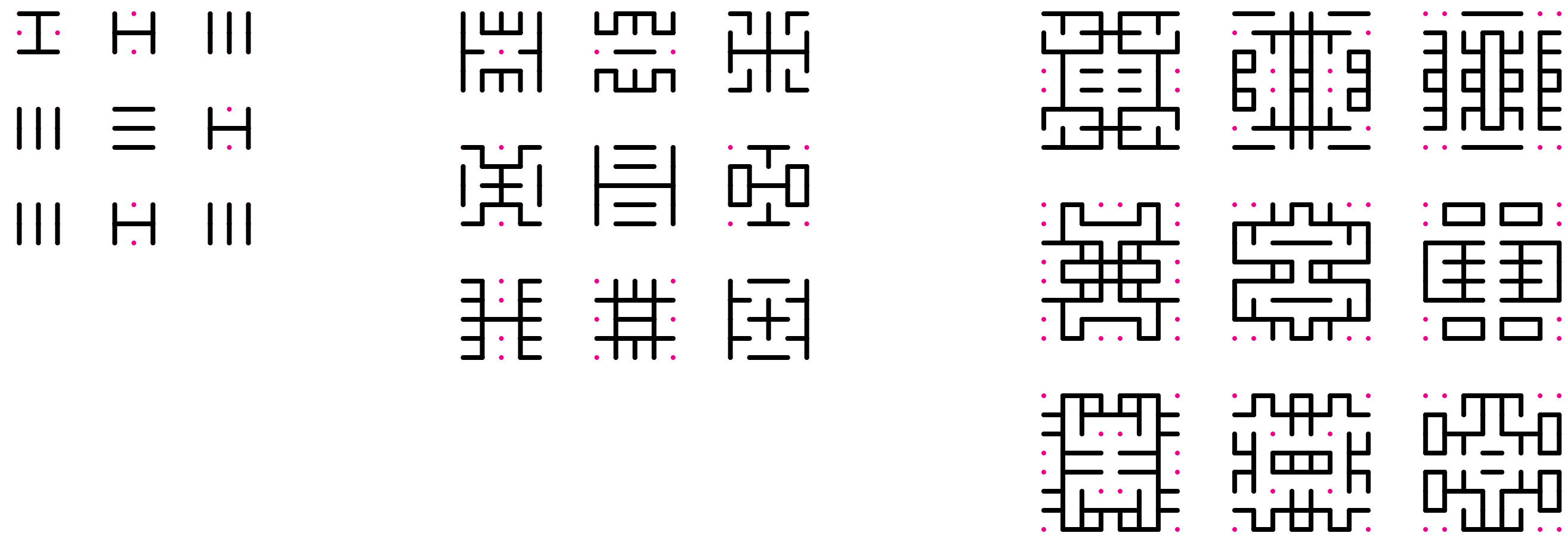




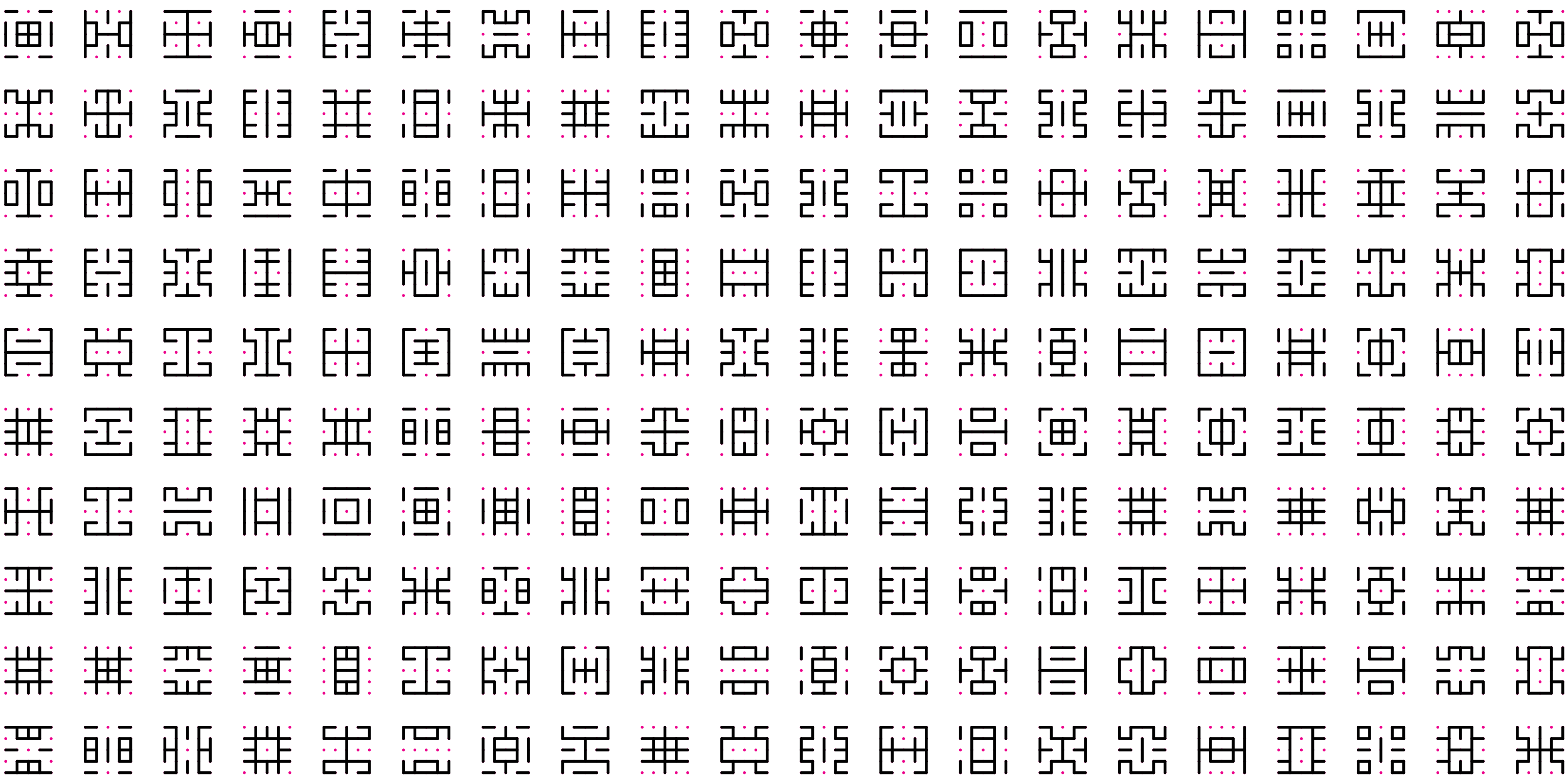
# Solution Space Modeling



# Generative Design—Grid Studies



# Generative Design—Grid Studies





# Generative Design—Grid Studies

## Rules:

- Horizontal symmetry
- Vertical symmetry
- 50% line utilization (rounded down to nearest even number)

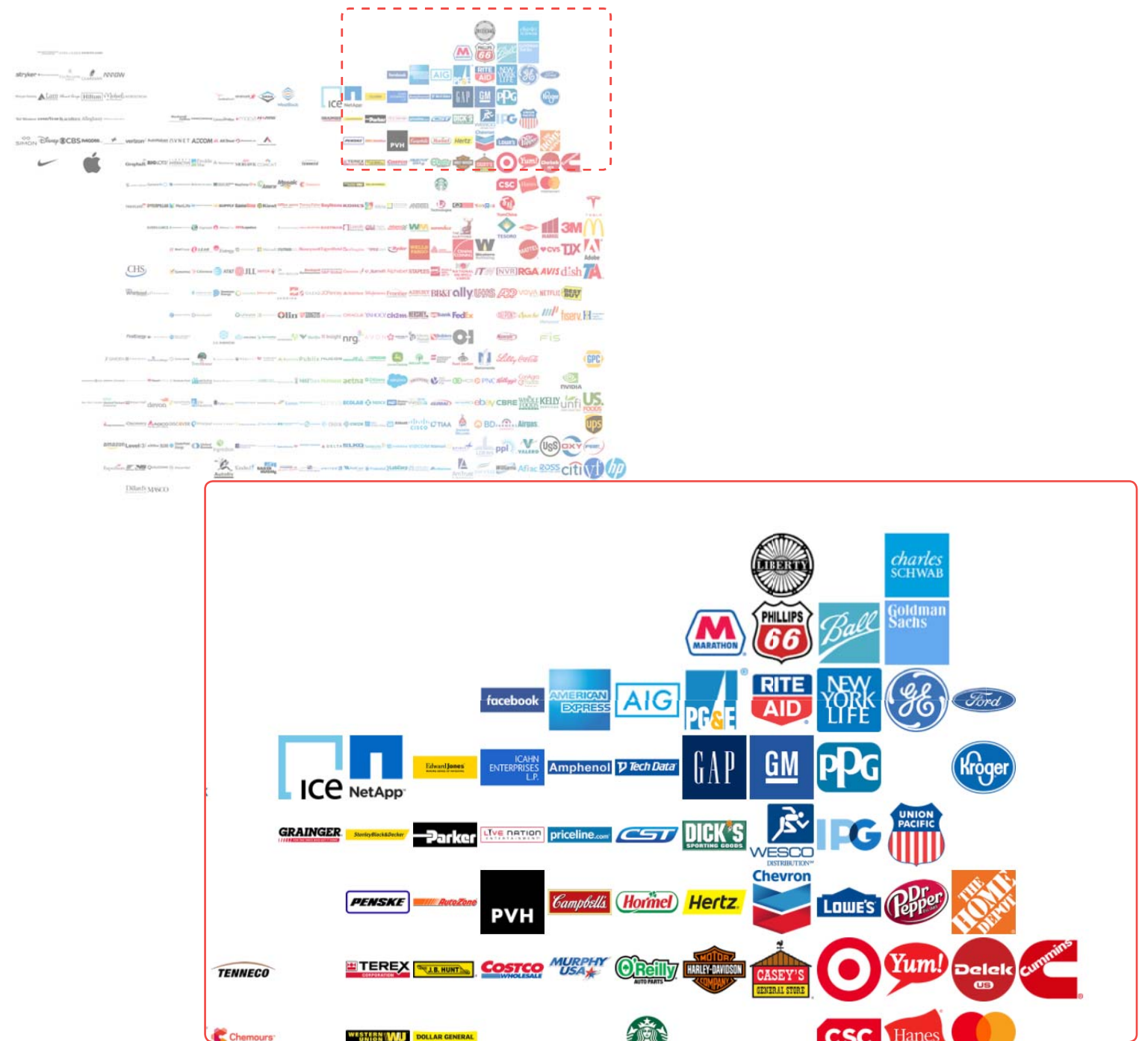
<https://github.com/knutsynstad/dots-and-lines>

# Generative Design—Fortune 500 Logos Grouped by Form and Color



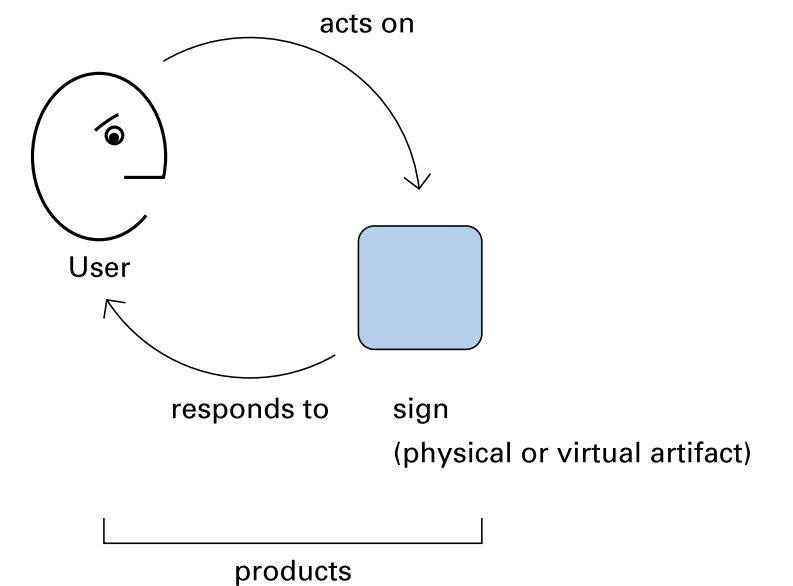


# Generative Design—Fortune 500 Logos Grouped by Form and Color



# Creating conditions in which others may design.

## User interacting with an artifact.

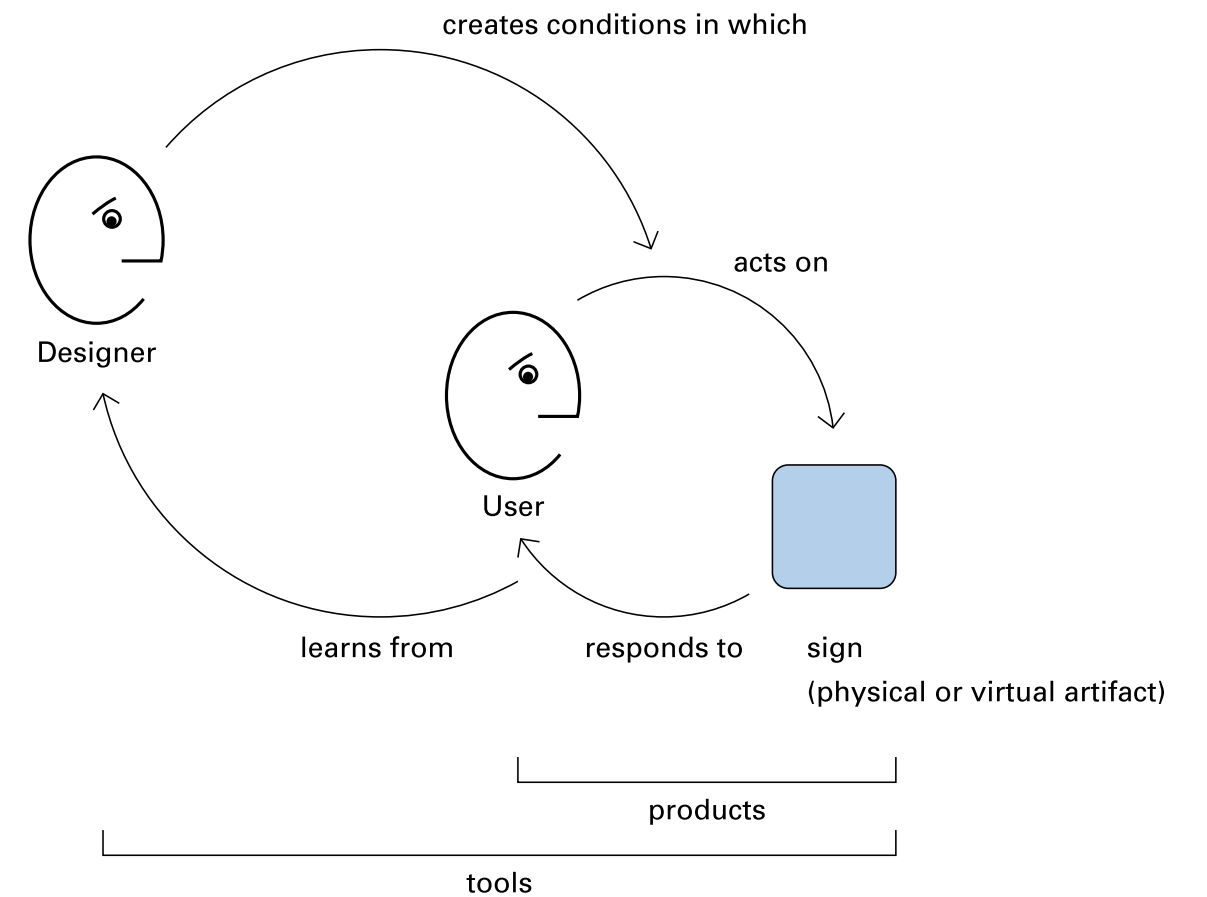




# Creating conditions in which others may design.

## Designer interacting with...

## User interacting with an artifact.

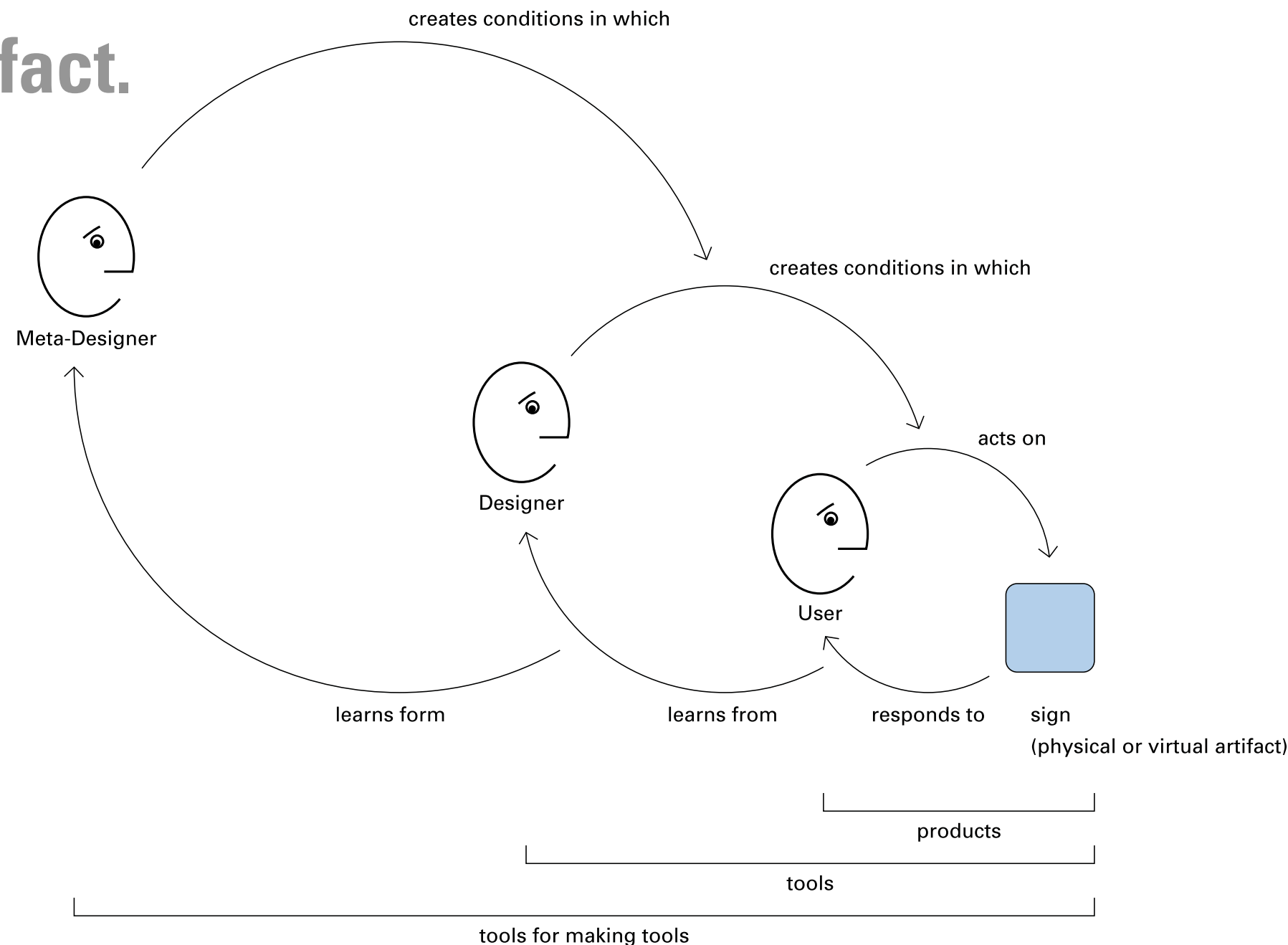


Creating conditions in which others may design.

**Meta-designer interacting with...**

**Designer interacting with...**

**User interacting with an artifact.**



*“[Winograd and Flores] go to the heart of the matter concerning design: ‘We encounter the deep questions of design when we recognize that in designing tools we are designing ways of being’.... ‘We create and give meaning to the world we live in and share with others.... we design ourselves (and the social and technological networks in which our lives have meaning) in language.’”*



— **Gui Bonsiepe**, *Interface an Approach to Design*, 1994 [115]  
(quoting from *Understanding Computers and Cognition*, 1986)



**Special thanks to**

**Anne Balsamo**

**Mihai Nadin**

**Knut Synstad**

**Jamie Ikeda**

**[hugh@dubberly.com](mailto:hugh@dubberly.com)**

**Presentation posted at**

**[www.dubberly.com/presentations/thirdera.pdf](http://www.dubberly.com/presentations/thirdera.pdf)**