Constraint-Based Design: A Gateway to AI

Edward Barnard

Constraint-Based Design: A Gateway to AI

The HPC Tradecraft Master Practitioner, Volume 3

Edward W. Barnard

This book is available at https://leanpub.com/constraint-design

This version was published on 2026-04-06

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Edward W. Barnard

HPC Tradecraft for Computer Scientists: What We Stopped Teaching
Teaching Mastery in the Classroom
High-Stakes Ethics
Francis Barnard Part A
Teaching Mastery in the Classroom
Unexpected Histories: Spotting Patterns and Making Connections That Others Miss
Large Language Model Architecture Patterns in PHP: No Mathematics Required
The Impossible Challenge Manual for Age 14 and Up, Even For Adults: How to accomplish what everyone says you can't
The Wizard's Lens: Learn to Think Like AI
Nobody but Us: A History of Cray Research's Software and the Building of the World's Fastest Supercomputer
Beyond Prompt Engineering
Billy Mitchell's Bombsight: Shaping the B-25 Mitchell Bomber

Contents

Chapter 1. User Experience

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Problem of Credibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Gateway to AI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Show Me The Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Layered Disorientation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Maze as Adversarial Crisis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Checkpoint System as Distributed Knowledge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Hall of Testing as Gated Progression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Scrap Heap as Consequence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Survival Requirements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Chapter 2. Developer Experience

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

The Case Method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Bare Metal Case

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Which Part is Relevant

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Work in Progress

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Road Map

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/constraint-design.

Appendix A. HPC Tradecraft Road Map

What is "HPC tradecraft" as I know it? This is high-performance computing (HPC) arising from signals intelligence (code breaking) needs. My particular lineage begins with the "Tunny" code breaking project at Bletchley Park during World War II. That produced such an odd-looking contraption that the "Wrens" operating it named it the Heath Robinson. W. Heath Robinson was a cartoonist known for portraying fantastically complex machines designed to accomplish simple tasks.

When U.S. Navy veterans built something similar for a similar purpose, they named it GOLDBERG in honor of cartoonist Rube Goldberg's fantastically complex machines, explicitly acknowledging the Heath Robinson tradecraft origins. One of the founders of Engineering Research Associates (ERA) was Bill Norris. ERA built GOLDBERG, and also hired Seymour Cray. Norris later founded Control Data Corporation (CDC) and Cray soon followed as their youngest employee. Cray later founded Cray Research, which I joined early 1980.

Thus the high-performance tradecraft I learned within Cray Research already had a 40-year history. Perhaps because so much was classified Top Secret, we never wrote down *how* we did things. Institutional knowledge was passed from person to person as mentorship, collaboration, and informal apprenticeship.

The 1995 Barrier

Circa 1995, something shifted. We hid the messy details behind abstractions, allowing for far more complex software development. We no longer needed to think of systems as a whole, and how things performed on bare metal.

For high-performance computing, this presents a problem. HPC tradecraft remains constraint-based design, but we can no longer see the constraints informing the design. This became a niche topic rarely covered in school (or the workplace).

Meanwhile, the pre-1995 tacit knowledge is retiring out of the workforce. My role as Custodian (the Russian term is more specific, хранитель) is to share this knowledge with you as a living practice.

Two Series

I created two book series to teach HPC tradecraft.

- "HPC Tradecraft Apprenticeship", three books, is the institutional tradecraft as I learned and practiced, and still practice. Tradecraft is useless unless replicable. The first book teaches my replication method; the second teaches *how* we accomplished what we did; the third teaches systems thinking, with close observation and characterization of HPC systems under load, using modern AI as the example. The three books are transmission protocol; HPC design; HPC operations.
- "The HPC Tradecraft Master Practitioner", three books, is the author letting loose and having fun, expressing his mastery of HPC tradecraft. High-Stakes Ethics prepares for the insurmountable power imbalance of a new graduate in the amoral corporate environment. Unexpected Histories demonstrates debugging skills applied outside computer science, showing their value and timeless nature. Constraint-Based Design demonstrates 1986 tradecraft directly applies to modern AI transformer design and usage.

Book 1. HPC Tradecraft for Computer Scientists: What We Stopped Teaching

I iteratively created book 1 as I circled what I actually have to share, why it is important, and how to make it accessible to the right persons.

- Show my teaching method of formation text that performs what it teaches, which I frame as the manner of replication.
- Act as filter and qualification, so that readers have the necessary information before buying/committing to the two large flagship books.
- State *Nobody but Us* as prerequisite to *Wizard's Lens*, which readers might perceive as problematic if they came for AI.

The sequence itself is formation. This was the teaching order within Cray Research.

Book 2. Nobody but Us: A History of Cray Research and the Building of the World's Fastest Supercomputer

This book is by far the largest and highest writing quality. However, persona walk-throughs consistently report the quality as uneven. This is because the book attempts to overcome the Edwin Abbott Flatland paradox. "Look here" in the midst of formation text appears to be uneven quality requiring an editor, and a human might see it the same way, but "look here" is pointing to something most readers do not have the vocabulary to see on their own.

The book serves a crucial purpose of credentialing the author (via demonstration not assertion). Why? Because the next book makes apparently-outrageous claims which are not proven until reading through to the final chapter. The claims are based in "Nobody but Us" tradecraft.

What Cray Research accomplished is well documented. This book teaches *how* we did it. But there is another aspect, which is the next book.

Book 3. The Wizard's Lens: Learn to Think Like AI

Think Jay Forrester and Donella Meadows and Eli Goldratt and Robert Gagné and Anders Ericsson, all of whom are referenced in the book. This is the craft of closely observing and characterizing HPC systems, using AI as the working example. This is the Cray Research attitude of treating barriers as opportunities. Barriers point to the point of maximum leverage. Meadows lists points of leverage. For my purposes, I collapse her list to Liebig's Law of the Minimum (plant growth is bounded by the relatively scarcest nutrient). This principle matches Goldratt's *The Goal*.

Rather than the skills in the prior book, this teaches shifting perspective, systems as a whole, interactions between systems at the boundary, and the attitude of not being distracted by the fact that it has never been done before.

This ends the primary series, the HPC Tradecraft Apprenticeship.

Book 4. High-Stakes Ethics

The book is a work in progress. My premise is that we do not give computer scientists, particularly recent college graduates, the skills to survive the corporate power imbalance. I wrote two essays regarding that power imbalance. They are the raw material from which I plan to write the book, but the essays themselves have urgent importance, thus publishing as a work in progress.

Book 5. Unexpected Histories: Spotting Patterns and Making Connections That Others Miss

This book consists of 15 essays/chapters, 49,000 words, demonstrating tradecraft in primary source analysis and constructing interesting narratives by "connecting the dots" that others miss. While nothing in here is technically difficult, some seem to have this aptitude and some do not. Book 2 has a chapter on what William Friedman has to say about developing intuition. This book is practice in developing that premise.

Tradecraft demonstrated across the centuries and outside computing demonstrates invariants, skills non-specific to a given technology, while demonstrating Piotr Galperin's final stage of formation.

Book 6. Constraint-Based Design: A Gateway to AI

In 1986, I wrote a bare metal device driver inside the Cray I/O Subsystem as a joke and to demonstrate the author's capability. It is a text adventure game, Swiss Adventure, inspired by Colossal Cave Adventure. What is useful is its characteristics.

The game itself forces the user (adventurer) into an operational apprenticeship, working with incomplete information in an unfamiliar context. I created a PHP/Laravel/React.js version that is live on my website, stateless, requiring no login or registration or data collection. The live version is the game itself presented as typing at the operator console, with side displays (essentially event traces) showing flow through the assembly language overlays, and flow through the LLM attention mechanism patterns.

Thus it is an executable companion to the book. The 1986 source code and the PHP engine are in a public GitHub repository, with the most recent timestamp in the repository as May 2019, which is well before GPT and Claude became well known publicly.

- The live website is here: Swiss Adventure
- The assembler and PHP source code are here: SwissAdventure

There are several aspects of interest.

UPDATE

The source code is in UPDATE format. Secondary sources (wikipedia) indicate CDC UPDATE is one of the earliest known source code management systems, from late 1950s or early 1960s, punch card based. I know from personal (as primary source) knowledge that the first Cray Research software person was hired directly from CDC. I know from internal statements that CDC's UPDATE was ported to Cray Research. While onsite at Boeing Computer Services 1980-1982, the front end systems were CDC Kronos. I used CDC UPDATE and Cray Research Update side by side. Thus I am clear on the provenance.

Relatively few extant examples of live code in UPDATE format exist. This one is publicly available on GitHub.

APML

When the CRAY-1/S was announced with a new I/O Subsystem (IOS) in addition to the mainframe CPU, the IOS had been developed as the "A" Processor. Extant documentation as of 1980 called it that. The cpu assembler was CAL, Cray Assembly Language. The IOS assembler was called APML, "A" Processor Macro Language.

The 1986 source code is a live, complete, package, and likely the only substantial example of IOS programming written by an actual Cray Research IOS programmer. It was written (as a student) during a class *teaching* IOS programming, but I was later part of the IOS group for 10 years, so can act as primary source asserting it is authentic and representative.

"Live" might be misleading in that no system still exists to run it. But the precise port to PHP *is* live and online.

Adventurer Experience

The adventurer experience has characteristics similar to many realms of operational training. I intentionally wrote it inspired by Colossal Cave Adventure and Zork. It might be worth noting that the authors of those two programs worked in operational environments. Thus I was not inventing the formation aspect. That was already a characteristic of text adventure games.

Constraint-Based Design

The source code has many levels and aspects of constraint-based design, and it is possible to teach the design without dwelling on assembler syntax. I implemented the training data (all navigation and text content) as a domain-specific language using the assembler macro and micro facility. I had a deep enough understanding (including making mods to the CAL assembler source) to do this. Adding game features did not require executable code changes.

Here is the odd part: The source code architecture is isomorphic to 2017 "attention is all you need" architecture function. Similar constraints shape similar solutions (Genrich Altshuller's TRIZ principle). My operating environment was quite similar to a hot LLM token context under load. Attention heads, weighting, pulling from external source then deallocating… quite a few features are present.

A working bare-metal example of constraint-based design can be useful. But a 1980s demonstration that HPC methods are relevant to modern AI has interesting implications, which is why I decided to include this as a book.