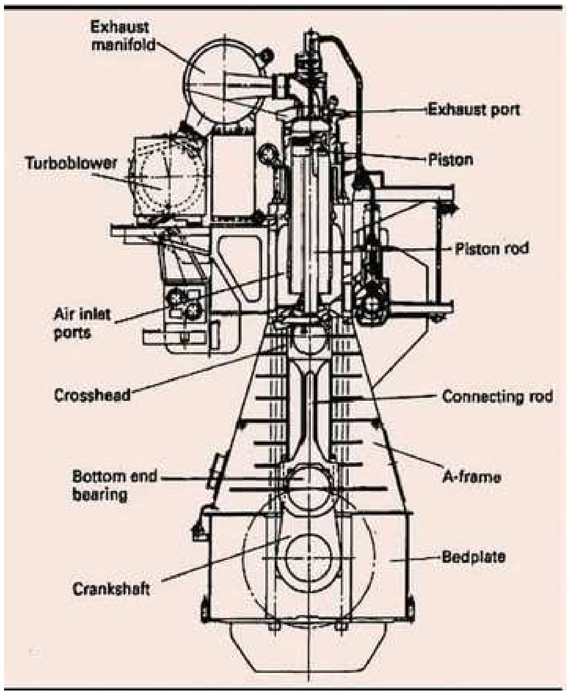
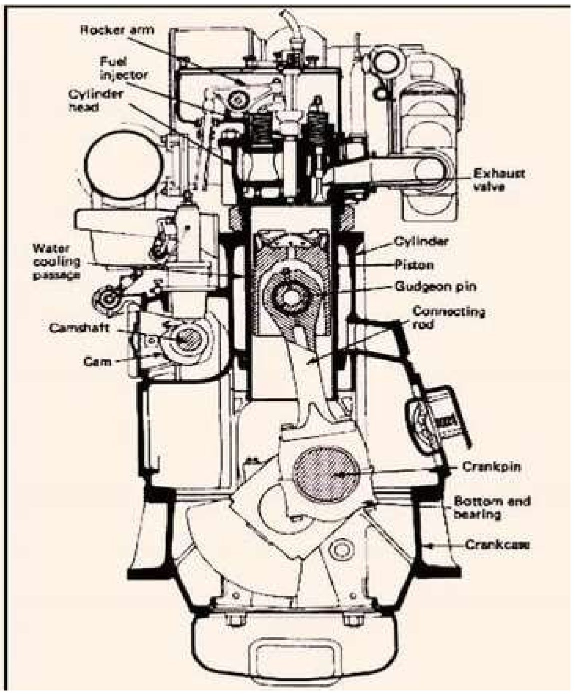
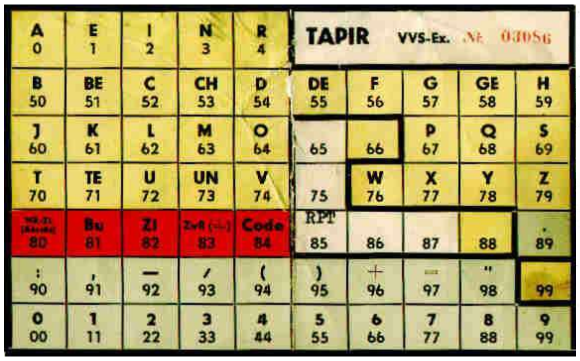
Systems I find elegant, and what they teach me.

Increasingly, the systems I find the most elegant are the ones where a small number of parts or principles provide disproportionate value. That is why I prefer the two-stroke engine to the 4-stroke: same power, half the displacement, easier to maintain. When solving problems, I like to first consider if there is a two-stroke solution.

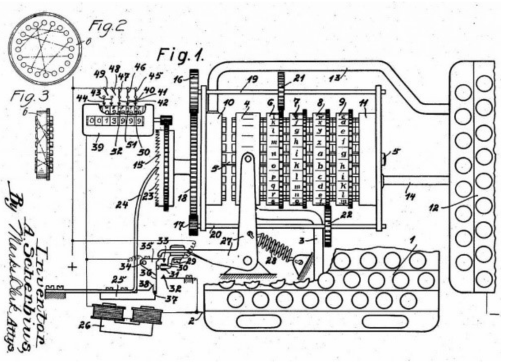




Similarly, the most impactful tools for end-users need not be complex. A good example of this is in cryptography, an area of research that is notoriously complex and often intentionally obtuse. And yet, the One-time Pad (1917), which encrypts messages using nothing more than a random number generator and simple addition, remains the only provably unbreakable code when used correctly. By contrast, later systems like the Enigma machine in WWII were mechanically intricate, with multiple rotors and plugboards, but that complexity hid structural weaknesses that were eventually exploited.

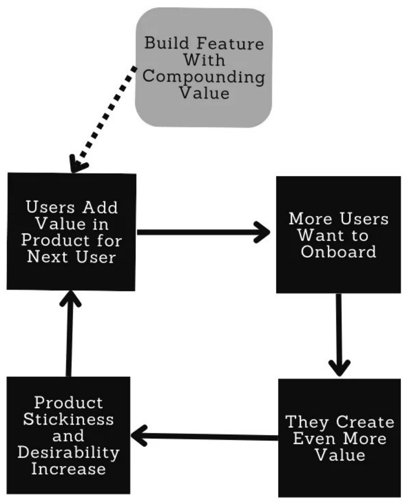


A 0	E 1	I 2	N 3	R 4	TAPIR VVS-Ex. № 03056				
B 50	BE 51	C 52	CH 53	D 54	DE 55	F 56	G 57	GE 58	H 59
J 60	K 61	L 62	M 63	O 64	65	66	P 67	Q 68	S 69
T 70	TE 71	U 72	UN 73	V 74	75	W 76	X 77	Y 78	Z 79
[illegible] 80	Bu 81	ZI 82	ZwR (+-) 83	Code 84	RPT 85	86	87	88	. 89
: 90	, 91	— 92	/ 93	(94	) 95	+ 96	= 97	" 98	99
0 00	1 11	2 22	3 33	4 44	5 55	6 66	7 77	8 88	9 99



My point is that I think elegance and complexity are too often conflated. While the one-time pad system may not have been a product fit for large-scale military use, this earlier innovation reminds me to focus on finding the most direct sequence of actions that solves an end-user's problem. What <u>wins</u>? What <u>works</u>? Focus on nothing else.

Market domination of a product, however, does not allow for an engineering mindset that focuses *solely* on short-term wins. We want to solve customer problems, but in a way that compounds the value of the service. How I conceptualize this is through what we might call the "nth User Value Engine." 1->n rather than 0->1.


Build Feature With Compounding Value
Users Add Value in Product for Next User
More Users Want to Onboard
They Create Even More Value
Product Stickiness and Desirability Increase

Practically, it's a system where builders focus not only on providing value but capturing it. This is done by building features that leave the next user better off because of what the last user did. As more customers are on board, they add their own structured data, feedback, and edge cases, which makes the engine more valuable, compounded further by network effects. Over time, you don't just have a list of logos; you have an engine that makes every new deployment make the whole solution line smarter, stickier, and easier to sell.

I don't find this system as elegant as the two-stroke, but I do find it useful.