The Architecture of Advantage: A Christensen Framework for Deciding Between Proprietary (In-House) and Modular (COTS) Software

Executive Summary

The "build vs. buy" software decision is a false dichotomy that has cost organizations trillions in misplaced resources and lost competitive advantage. Framed as a simple financial or technical choice, it is most often made at the wrong level, by the wrong people, and for the wrong reasons. This report reframes the decision as the most critical output of a firm's corporate strategy: It is a fundamental architectural choice about *where* in the value chain a company must be proprietary and interdependent to win, and *where* it must be modular and generic to be efficient.

This framework moves the decision from a Total Cost of Ownership (TCO) calculation to a strategic diagnosis. It begins not with the software's features, but with the end customer's "Job to Be Done" (JTBD). We demonstrate that this is the only logical starting point. By synthesizing multiple theories from the Christensen Institute—from Jobs to Be Done and the Theory of Interdependence and Modularity to the Law of Conservation of Attractive Profits, Disruptive Innovation, and the Tools of Cooperation—this report provides a robust model for leaders to identify the "bottleneck" in their value chain where they *must* build a proprietary, interdependent advantage.

We argue that "buying" (a modular Commercial Off-the-Shelf, or COTS, solution) at this strategic bottleneck is an act of strategic surrender, commoditizing a firm's core advantage.[1] Conversely, "building" (an interdependent system) in a non-strategic, "good enough" area is a catastrophic waste of resources. This report provides the diagnostic framework for making this architectural decision correctly and concludes with a toolkit for executives to manage its profound, long-term consequences.

Part 1: The Strategic Foundation - Starting with the End Customer

1.1. The "Job to Be Done" as the Unit of Strategic Analysis

The conventional approach to software decisions begins with an internal feature list—a process that is fatally flawed. It optimizes a *solution* before understanding the *problem*. A strategic framework must begin with the only stakeholder that matters: the end customer.

The Christensen Institute's Jobs to Be Done (JTBD) Theory posits that customers do not simply buy products or services; they "hire" them to make "progress" toward a goal within a specific "circumstance".[2] This "job" is the true unit of strategic analysis.

This job is not merely functional; it is a complex bundle of three forces [4]:

1. **Functional:** The practical task to be accomplished (e.g., "grow my financial assets").
2. **Emotional:** How the customer wants to feel (e.g., "feel secure about my family's future").[6]
3. **Social:** How the customer wants to be perceived by others (e.g., "be seen as a responsible provider").[7]

This distinction creates an immediate "analysis gap." Most COTS software is designed to address the *average* company's *average* functional job (e.g., a CRM to "manage a sales pipeline").[8] By its very nature, it is structurally incapable of solving a *specific* company's unique emotional or social JTBD.

The decision of whether to build or buy a Customer Relationship Management (CRM) system, therefore, depends entirely on the end customer's JTBD. If the customer's job is purely functional ("Help me buy a basic product at the lowest price"), the relationship itself is a commodity. If the job is highly emotional and social ("Help me feel like a valued, high-status professional"), the relationship *is* the product, and a generic COTS solution is, by definition, insufficient.[1]

1.2. From Customer Job to Corporate Value Chain: Mapping the

Architecture of Delivery

Once the end-customer's JTBD is defined, the organization must map how its internal value chain delivers that "progress." The "build vs. buy" decision is a choice about which *link* in this chain will be the firm's proprietary weapon.

The customer's JTBD can be deconstructed into a "Job Map," which outlines the universal steps a customer takes to get a job done: Define, Locate, Prepare, Confirm, Execute, Monitor, Modify, and Conclude.[9] The company's internal value chain is the operational mirror of this map, and the greatest value is created at the steps where the customer "struggles" the most.[9]

The "build vs. buy" decision forces a company to identify this single, critical link. Not all links in the value chain are created equal.

- For a **discount e-commerce** customer (JTBD: "lowest price, fast"), the key Job Map steps are "Execute" (fast checkout) and "Monitor" (reliable tracking). The company's differentiating link is its supply chain and pricing engine.
- For a **private wealth** client (JTBD: "feel secure"), the key Job Map steps are "Define" (plan approach), "Monitor" (check on progress), and "Modify" (adjust to life events). The company's differentiating link is proactive, personalized communication.[10] The CRM is central to this.

A "build everything" or "buy everything" strategy is always wrong. The correct strategy is to create a hybrid architecture: "buy" (modularize) every link of the value chain *except* for the single link that is critical to the JTBD. At that one link, the company must "build" (integrate) a proprietary, interdependent system.

1.3. Locating the "Bottleneck": Applying the Law of Conservation of Attractive Profits

This framework explains *where* in the value chain profit and power will migrate, and therefore, *where* a company must build its proprietary advantage.

The "Law of Conservation of Attractive Profits" (also called the Law of Conservation of Modularity) states that as one part of a value chain becomes modular and commoditized (e.g., PC hardware), the profits and power migrate to the adjacent, interdependent layer (e.g., the operating system).[12]

Profits always flow to the "bottleneck"—the point in the value chain that is "not yet good enough" to satisfy the end customer's needs.[14] The company that can create an *integrated, interdependent* system to solve this specific bottleneck will capture the industry's profits.[16]

This law transforms the "build vs. buy" question into: "At which link in our value chain is performance *not good enough* to meet the JTBD, and therefore, where must we create a proprietary, interdependent system to capture the industry's profits?"

Applying this to the CRM:

1. **Past:** In the 1990s, enterprise software itself was the bottleneck. Performance was "not good enough." Companies *had* to build custom, interdependent CRMs.
2. **Present (Modular):** COTS/SaaS solutions like Salesforce emerged, *modularizing* the CRM function.[17] For most businesses, these modular solutions are now "good enough."
3. **Present (Interdependent):** For the private wealth firm, the "good enough" COTS CRM is the *new* bottleneck. It is "insufficient" because its modularity *prevents* the deep personalization required by the customer's emotional JTBD.

This reveals the central warning of this report: If a company's differentiating bottleneck *is* the customer relationship, and it chooses to "Buy" a modular COTS CRM, it is *actively ceding the single most profitable, powerful point in its value chain to a third-party vendor*.[1] It is an act of strategic surrender.

Part 2: The Decision Framework - Interdependence vs. Modularity

2.1. A Strategic Re-framing: The "Build vs. Buy" Decision as an Architectural Choice

The terms "build" and "buy" are simplistic. The real choice is between two different *architectures* defined by the Christensen Institute: Interdependent and Modular.[18]

- **Modular Architecture (Buy/COTS):** This architecture is chosen when components "fit and work together in well-understood and highly-defined ways".[19] This is possible only when the interfaces between components are **"specifiable, verifiable, and predictable"**.[20] A COTS software solution is the epitome of a modular component. It allows for flexibility, speed, and relies on "various independent providers".[20]

- **Interdependent Architecture (Build/In-House):** This architecture is *required* when the "way one part is designed and made depends on the way the other is designed and made".[20] When performance is "not good enough," the interfaces are *not* specifiable or predictable.[22] The only way to optimize for a massive leap in performance is to control the *entire* system, tightly integrating all components.[19] This is a proprietary, in-house "build."

The decision, therefore, is not made for the whole company, but at each *interface* in the value chain.[20] Can an organization write a perfect specification (specifiable, verifiable, predictable) for its CRM's connection to its data warehouse? If yes, it can use a modular (COTS) solution. Can it write a perfect specification for how a wealth advisor's "empathy" (a core part of the JTBD) interfaces with the software? No. That interface is *not* specifiable, and therefore *must* be developed within an interdependent, integrated system where the human and the software are designed together.[15]

When a company "buys" a COTS solution, it is not just buying software; it is buying a *pre-defined, "best practice" definition of cause-and-effect* for a specific business process.[25] This is efficient *only if* that process is not a source of competitive advantage.

2.2. The Performance Threshold: When Is "Good Enough" Good Enough?

The concept of "overshooting" is the trigger that tells a company *when* to shift from an interdependent (build) architecture to a modular (buy) one.

"Overshooting" is a core concept in the Theory of Disruptive Innovation.[27] Incumbent firms, listening to their most demanding customers, relentlessly add performance (features, complexity, cost) to their products. Eventually, their products "overshoot" the needs of the mainstream market, which would be happy with a *simpler, cheaper, "good enough" product*.[28]

This "overshooting" by complex, custom-built internal legacy systems created the market for COTS and SaaS.[27] A "good enough" COTS solution enters at the low end, providing a simpler, more affordable modular alternative.[31]

This provides a clear litmus test for the build vs. buy decision:

- **When to Buy (Modular):** When the performance of COTS solutions has become "good enough" (or *more* than good enough) to satisfy the requirements of a *non-differentiating* link in the value chain.[15]
- **When to Build (Interdependent):** When the performance of *all* modular (COTS) solutions is "insufficient" to meet the unique demands of the *differentiating link* (the

"bottleneck").[23]

This process is cyclical. The "sea of sameness" [1] created when all competitors adopt the same COTS solution is a symptom of that solution itself "overshooting." This creates a *new* "not good enough" bottleneck (e.R., lack of true personalization) at a different layer, forcing a *new* "build" (interdependent) cycle—just as the Law of Conservation of Attractive Profits predicts.[34]

2.3. The Diagnosis: A Framework for Architectural Choice

The synthesis of these theories provides a clear diagnostic framework, summarized in Table 1. The decision is not a simple financial trade-off but a strategic choice about performance, architecture, and long-term risk.

Table 1: The Core Architectural Decision Framework

Dimension	**Build (Interdependent)**	**Buy (Modular)**
Governing Theory	Law of Conservation of Attractive Profits [14]	Theory of Disruptive Innovation [27]
Architectural Choice	Interdependent Architecture [20]	Modular Architecture [20]
Performance Goal	Optimize for unique performance (when COTS is "insufficient") [23]	Standardize for efficiency (when COTS is "good enough") [33]
Target Process	The core, differentiating "bottleneck" of the value chain [15]	A non-differentiating, "table stakes" link in the value chain [35]
Strategic Rationale	To create a proprietary, defensible asset and competitive advantage [1]	To leverage "best practice" efficiency and focus resources elsewhere [33]
Core Long-Term Risk	The Interdependence Trap (The Innovator's Dilemma /	The Modularity Trap (SaaS Sprawl / Process Rigidity) [38]

	Legacy System) [36, 37]	

Part 3: Practical Case Study - The Strategic CRM Decision

Applying the framework to the CRM decision reveals two starkly different, but equally correct, outcomes.

3.1. Scenario A: The "Modular" Company (e.g., Discount E-commerce)

- **End Customer's JTBD:** "Get my product reliably, quickly, and at the lowest price".[2] This is an overwhelmingly *functional* job.[4]
- **Differentiating Link (Interdependent):** The "bottleneck" [14] is not the customer relationship; it is the *supply chain optimization and algorithmic pricing system*. This is where performance is "not good enough" across the industry, and where this firm must build a proprietary, interdependent system to win.
- **Role of CRM:** The CRM is a *supporting, modular component*. Its "job" is to provide a standardized, "good enough" [27] interface for customer service (e.R., "track my order," "process a return"). A generic process is acceptable and efficient.
- **Decision: Buy (COTS)**. The company should choose a "good enough" modular solution (like Salesforce or HubSpot) and *integrate* it with its proprietary logistics and pricing engines.[33] Wasting elite engineering talent to "build" a custom CRM in this scenario would be a catastrophic misallocation of resources.

3.2. Scenario B: The "Interdependent" Firm (e.g., Private Wealth Advisory)

- **End Customer's JTBD:** "Feel financially secure through a deeply personalized and proactive relationship." This is an overwhelmingly *emotional* and *social* job.[5]
- **Differentiating Link (Interdependent):** The "bottleneck" [14] *is* the customer relationship engine itself. The ability to create a unique, personalized, high-touch experience is the *entire* business. The modular COTS solutions [32] are "insufficient" because they are built

for the *average* sales process, not for tracking complex family relationships, emotional milestones, and long-term trust.[1]

- **Role of CRM:** The CRM is *not* a supporting tool; it *is* the core differentiating system. It is the interdependent architecture that must be built.
- **Decision: Build (In-House)**. The firm must build a proprietary, interdependent system that tightly integrates client data, communication logs, financial planning tools, and human advisor workflows. This system *is* the competitive advantage. Buying a COTS solution here would be the "strategic surrender" discussed in Part 1.

Table 2: Case Study Summary - The Strategic CRM Decision

Dimension	**Scenario A: Discount E-commerce**	**Scenario B: Private Wealth Advisory**
End Customer's JTBD	"Get my product reliably, quickly, and at the lowest price."	"Feel financially secure through a deeply personalized and proactive relationship."
Primary Job Dimension	Functional [4]	Emotional & Social [5]
Differentiating "Bottleneck"	Supply Chain & Algorithmic Pricing	The Personalized Relationship Engine
Role of CRM	Supporting / Modular	Differentiating / Interdependent
Performance of COTS	"Good Enough" [33]	"Insufficient" [1]
Strategic Decision	**Buy (Modular)**	**Build (Interdependent)**

Part 4: Managing Long-Term Consequences and Risks

The "build vs. buy" decision is not an endpoint. It is the beginning of a new set of strategic challenges. Each path introduces a specific, symmetric, and dangerous long-term risk.

4.1. The Modularity Trap: The Risk of Buying and the "Chaos of Uncontrolled Growth"

Choosing "Buy" (modular) is not a "fire and forget" solution. It introduces the *Modularity Trap*, which manifests in two forms:

1. **"SaaS Sprawl":** This is the "Chaos of Uncontrolled Growth".[41] The ease of "buying" modular COTS solutions leads to an "unchecked growth" and "uncontrollable growth" of applications within an organization.[41] This creates a cascade of risks:
    - **Financial Risk:** Duplicate spending on redundant applications and wasted spending on unused licenses.[38]
    - **Security Risk:** A proliferation of unvetted applications, data privacy breaches, and the rise of "Shadow AI" as employees procure their own tools.[38]
    - **Operational Risk:** Massive data silos, poor cross-functional collaboration, and "integration chaos" as IT struggles to connect disparate systems.[38]
2. **"Process Rigidity":** This is the more insidious risk. The COTS solution, built for the "average" process [39], forces the company to *conform its processes to the software*. This "one-size-fits-all" model [32] creates "customization bottlenecks" and "rigidity".[39] The initial promise of COTS agility [45] becomes a long-term weakness, as the company becomes locked into the vendor's release cycles and process logic, making it *slower* to respond to market changes.

4.2. The Interdependence Trap: The Risk of Building and Sowing the Seeds of Disruption

Choosing "Build" (interdependent) is not a permanent solution. It introduces the *Interdependence Trap*, which is a perfect articulation of "The Innovator's Dilemma".[36]

A successful interdependent system (the custom-built CRM) inevitably becomes a "legacy system".[37] It was built perfectly for *yesterday's* problem. This happens because the company's business model—its "Key Processes" and "Profit Formula"—*solidifies* and becomes "increasingly interconnected" around this successful proprietary system.[49]

This system, once a source of advantage, becomes a "blocking resource".[51] It is costly to maintain, creates new data silos, and is often run by the few remaining developers who understand its "obsolete programming language".[37]

The company is now *blind* and *incapable* of adopting a new, simpler, disruptive technology (like a modular COTS solution). Its entire "Profit Formula" and "Processes" are hard-coded to the old system.[50] It "does everything right" by listening to its *existing* high-value customers, and in doing so, it fails to deliver a "new value proposition".[49] The proprietary system, by being constantly improved, *becomes* the "overshot" product [15] that creates the market opening for a "good enough" modular COTS solution to attack from the low end.

4.3. Leading the Implementation: Applying the Tools of Cooperation

The "build" or "buy" choice dictates the *type* of change management required. Using the wrong tool guarantees failure. The Christensen Institute's "Tools of Cooperation" framework [54] provides the correct approach, based on two axes: 1) Agreement on "Goals" (What we want) and 2) Agreement on "Cause and Effect" (How to get it).[55]

- **Implementing "Buy" (COTS):**
  - **The Situation:** When a company "buys" a COTS solution, it is *buying the "cause and effect"*.[25] The software dictates the "how" (e.g., "this is the new sales process"). There is *high agreement on cause and effect* (it's embedded in the tool). However, there is often *low agreement on goals* (the sales team doesn't want to change *their* process; their goals conflict with IT's goals).
  - **The Correct Tool: Management Tools**.[55] The implementation must be "coordinative and process-oriented".[56] This requires **training, standard operating procedures (SOPs), and measurement systems** [57] to force alignment and drive adoption of the new, pre-defined process.
- **Implementing "Build" (Interdependent):**
  - **The Situation:** When a company "builds" a proprietary system, the team is united by a common, aspirational goal (e.g., "We must create the world's best wealth management experience"). There is *high agreement on goals*. However, the "how" is completely unknown. There is *low agreement on cause and effect* because the team is *discovering* the solution as it goes.[59]
  - **The Correct Tool: Leadership Tools**.[55] The implementation must be "results-oriented".[57] Management Tools (like a rigid SOP) will kill the project. Instead, the leader must use **vision, charisma, and salesmanship** [57] to inspire the team to "just go out and do it" [57] and navigate the uncertainty of discovery.

This framework explains why so many software projects fail. Managers "Manage" (using Management Tools) when they should be "Leading" (using Leadership Tools). They try to apply rigid processes to an innovative "build" project, crushing its potential. Conversely, they try to "Lead" (with vision) a COTS implementation, when they should be "Managing" (with training and SOPs), resulting in chaos and low adoption.

Part 5: Conclusion and The CEO Toolkit for Architectural Advantage

5.1. The Architecture of Advantage: A Summary of the Thesis

The "build vs. buy" decision is not a decision. It is a diagnosis. It is the single most important, recurring strategic output of an organization's leadership. It is the act of reading the market (JTBD), understanding the flow of profits (Conservation of Profits), and then consciously choosing *where* in the value chain to place an architectural bet. The choice is where to be generic (modular/buy) to be efficient, and where to be proprietary (interdependent/build) to win. This is the act of designing a firm's "Architecture of Advantage."

5.2. A Diagnostic Tool for Leaders (The CEO Toolkit)

To make the right architectural choice, leaders must ask these five questions in this specific order.

1. **The Customer Question (JTBD):** What is our end customer's *true* Job to Be Done, including its functional, emotional, and social components? [2]
2. **The Value Chain Question (Bottleneck):** To deliver that JTBD, at which *one link* in our value chain must we be proprietary and exceptional? Where is the "not good enough" bottleneck where industry profits are migrating? [14]
3. **The Triage Question (Architecture):** Is this specific software project aimed at that *one differentiating bottleneck* or is it a *supporting component*?
4. **The "Buy" (Modular) Path:** If the software is a *supporting component*, are the available COTS (modular) solutions "good enough" to execute the task? [23]
   - **(If Yes) -> DECISION: BUY.** Leverage a modular solution. Use **Management Tools** (training, SOPs) to implement.[58] Set a long-term strategy to manage "SaaS Sprawl"

and "Process Rigidity".[38]

5. **The "Build" (Interdependent) Path:** If the software *is* the *differentiating bottleneck*, are the COTS (modular) solutions "insufficient" to deliver our unique, proprietary advantage?[23]
   - **(If Yes) -> DECISION: BUILD.** Commit to an interdependent architecture. Use **Leadership Tools** (vision, charisma) to implement.[57] Set a long-term strategy to manage the "Innovator's Dilemma" and prevent the system from becoming a "legacy" trap.[36]

Obras citadas

1. Unlocking the Strategic Power of Build vs. Buy: The 6-Factor Framework, fecha de acceso: noviembre 6, 2025, https://www.baytechconsulting.com/blog/build-vs-buy-strategic-framework-2025
2. Jobs to Be Done Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/theory/jobs-to-be-done/
3. Jobs to Be Done Resources - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/resources/theory/jobs-to-be-done/
4. The Core Tenets of Jobs-to-be-Done Theory | by Tony Ulwick, fecha de acceso: noviembre 6, 2025, https://jobs-to-be-done.com/the-5-tenets-of-jobs-to-be-done-theory-ba58c3a093c1
5. What is Jobs to Be Done? Here's Our Playbook | Strategyn, fecha de acceso: noviembre 6, 2025, https://strategyn.com/jobs-to-be-done/jobs-to-be-done-playbook/what-is-jobs-to-be-done/
6. fecha de acceso: noviembre 6, 2025, https://strategyn.com/jobs-to-be-done/jobs-to-be-done-playbook/what-is-jobs-to-be-done/#:~:text=02.-,Jobs%20are%20functional%20%E2%80%94%20with%20emotional%20and%20social%20components,their%20peers%2C%20friends%20or%20others.
7. Understanding the 3 types of jobs to be done - LogRocket Blog, fecha de acceso: noviembre 6, 2025, https://blog.logrocket.com/product-management/3-types-of-jobs-to-be-done/
8. Jobs to Be Done Framework: A Guide for Product Teams - Product School, fecha de acceso: noviembre 6, 2025, https://productschool.com/blog/product-fundamentals/jtbd-framework
9. Jobs-to-be-Done | A Comprehensive Guide - Strategyn, fecha de acceso: noviembre 6, 2025, https://strategyn.com/jobs-to-be-done/
10. Build vs. buy - A strategic framework for evaluating third-party solutions - Thoughtworks, fecha de acceso: noviembre 6, 2025, https://www.thoughtworks.com/content/dam/thoughtworks/documents/e-book/tw_ebook_build_vs_buy_2022.pdf

11. A multi-layered approach to CRM implementation: An integration perspective | Request PDF, fecha de acceso: noviembre 6, 2025, https://www.researchgate.net/publication/222275568_A_multi-layered_approach_to_CRM_implementation_An_integration_perspective
12. Conservation of attractive profits - Applied Mathematics Consulting, fecha de acceso: noviembre 6, 2025, https://www.johndcook.com/blog/2009/06/22/conservation-of-attractive-profits/
13. Ten years ago: Clayton Christensen on Capturing the Upside - Asymco, fecha de acceso: noviembre 6, 2025, https://asymco.com/2014/06/23/clayton-christensen-on-capturing-the-upside/
14. Finding Power - Every, fecha de acceso: noviembre 6, 2025, https://every.to/divinations/finding-power-432187
15. Six Keys to Building New Markets by Unleashing Disruptive Innovation - Baker Library, fecha de acceso: noviembre 6, 2025, https://www.library.hbs.edu/working-knowledge/six-keys-to-building-new-markets-by-unleashing-disruptive-innovation
16. RIPPLES THROUGH THE VALUE CHAIN: HOW AN UPSTREAM INNOVATION SHAPES PROFIT AND SCOPE IN A SECTOR, fecha de acceso: noviembre 6, 2025, https://mackinstitute.wharton.upenn.edu/wp-content/uploads/2013/04/Jacobides-Michael-Veloso-Francisco-Wolter-Claudio_Ripples-through-the-Value-Chain-How-an-Upstream-Innovation-Shapes-Profit-and-Scope-in-A-Sector.pdf
17. 'CRM Disrupter in Cloud' with the lens of Clayton Christensen's disruptive strategy - Medium, fecha de acceso: noviembre 6, 2025, https://medium.com/@venus.pearl2000/crm-disrupter-in-cloud-with-the-lens-of-clayton-christensens-disruptive-strategy-8d72076447a4
18. Modularity Resources - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/resources/theory/modularity/
19. What is Modularity Theory? - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/video/what-is-modularity-theory/
20. Modularity Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/theory/modularity/
21. Infographic: Definitions & Conditions for Modularity Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/graphic/infographic-definitions-conditions-for-modularity-theory/
22. Why understanding Modularity Theory is key to market creation - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/blog/why-understanding-modularity-theory-is-key-to-market-creation/
23. Clayton M. Christensen, The Thought Leader Interview - Strategy+business, fecha de acceso: noviembre 6, 2025, https://www.strategy-business.com/article/14501
24. Design systems, modularity and interdependence - Ryan Singer, fecha de acceso: noviembre 6, 2025, https://www.ryansinger.co/design-systems-modularity-and-interdependence/
25. COTS vs. custom software for logistics and transportation - Adexin, fecha de

acceso: noviembre 6, 2025, https://adexin.com/blog/custom-vs-off-the-shelf-software-for-logistics/

26. COTS vs. Custom: DoD Software Strategies in Action - COTS Journal, fecha de acceso: noviembre 6, 2025, https://www.cotsjournalonline.com/cots-vs-custom-dod-software-strategies-in-action/
27. Disruptive innovation - Wikipedia, fecha de acceso: noviembre 6, 2025, https://en.wikipedia.org/wiki/Disruptive_innovation
28. Disruptive Innovation | The Encyclopedia of Human-Computer Interaction, 2nd Ed., fecha de acceso: noviembre 6, 2025, https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/disruptive-innovation
29. Foundations for Growth: How to Identify and Build Disruptive New Businesses, fecha de acceso: noviembre 6, 2025, https://sloanreview.mit.edu/article/foundations-for-growth-how-to-identify-and-build-disruptive-new-businesses/
30. Disruptive Innovation Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/theory/disruptive-innovation/
31. COTS vs aPaaS: When 80% isn't Good Enough - Appian, fecha de acceso: noviembre 6, 2025, https://appian.com/blog/2018/cots-vs-apaas-when-80-isnt-good-enough
32. Commercial Off-the-Shelf (COTS) Software vs Custom Development | Mendix, fecha de acceso: noviembre 6, 2025, https://www.mendix.com/blog/cots-software-vs-custom-low-code-development/
33. Build or Buy: Deciding between Custom Software or COTS - Spot Solutions, fecha de acceso: noviembre 6, 2025, https://spotsolutions.com/build-or-buy/
34. Netflix and the Conservation of Attractive Profits – Stratechery by ..., fecha de acceso: noviembre 6, 2025, https://stratechery.com/2015/netflix-and-the-conservation-of-attractive-profits/
35. The Innovator's Dilemma - Wikipedia, fecha de acceso: noviembre 6, 2025, https://en.wikipedia.org/wiki/The_Innovator%27s_Dilemma
36. What is a Legacy System? | Talend, fecha de acceso: noviembre 6, 2025, https://www.talend.com/resources/what-is-legacy-system/
37. Stop SaaS Sprawl: 4 Steps to Stymie Portfolio Growth - Zylo, fecha de acceso: noviembre 6, 2025, https://zylo.com/blog/saas-sprawl/
38. COTS Procurement Software Risks: Why Government Agencies ..., fecha de acceso: noviembre 6, 2025, https://appian.com/blog/2025/cots-procurement-software-risks-for-government
39. Infographic: 3 critical must-know's about Jobs to Be Done - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/graphic/infographic-3-critical-must-knows-about-jobs-to-be-done/
40. SaaS Sprawl: The Hidden Chaos In Your SaaS Stack | Zluri, fecha de acceso: noviembre 6, 2025, https://www.zluri.com/blog/saas-sprawl

41. What Is SaaS Sprawl and What Can You Do About It? - FinQuery, fecha de acceso: noviembre 6, 2025, https://finquery.com/blog/saas-sprawl/
42. SaaS sprawl is a rising concern for IT: Causes, challenges, and best practices | BetterCloud, fecha de acceso: noviembre 6, 2025, https://www.bettercloud.com/monitor/saas-sprawl-is-a-rising-concern-for-it-causes-challenges-and-best-practices/
43. What Is SaaS Sprawl and How To Control It - Torii, fecha de acceso: noviembre 6, 2025, https://www.toriihq.com/articles/saas-sprawl-control
44. What Is COTS Software? A Simple Guide - AppsRhino, fecha de acceso: noviembre 6, 2025, https://www.appsrhino.com/blogs/what-is-cots-software
45. The Confusion Around COTS | BioPharm International, fecha de acceso: noviembre 6, 2025, https://www.biopharminternational.com/view/confusion-around-cots
46. How & When to Modernize Legacy Systems - Hyland, fecha de acceso: noviembre 6, 2025, https://www.hyland.com/en/resources/articles/legacy-system-modernization
47. Legacy Software: The Ball and Chain Holding your Business Back - CPGvision, fecha de acceso: noviembre 6, 2025, https://www.cpgvision.com/blog/legacy-software-the-ball-and-chain-holding-your-business-back
48. Beyond the profit formula: The business model to drive health ..., fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/blog/beyond-the-profit-formula/
49. Business Model Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/theory/business-models/
50. Compatibility Issues with Legacy Systems During Upgrades - Visvero, fecha de acceso: noviembre 6, 2025, https://visvero.com/compatibility-issues-with-legacy-systems-during-upgrades/
51. The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail, fecha de acceso: noviembre 6, 2025, https://cpcglobal.org/publications/The%20Innovators%20Dilemma.pdf
52. What Is Disruptive Innovation Theory? 4 Key Concepts - HBS Online, fecha de acceso: noviembre 6, 2025, https://online.hbs.edu/blog/post/4-keys-to-understanding-clayton-christensens-theory-of-disruptive-innovation
53. Tools of Cooperation Resources - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/resources/theory/tools-of-cooperation/
54. Tools of Cooperation Theory - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/theory/tools-of-cooperation/
55. Infographic: When to use the Tools of Cooperation - Christensen Institute, fecha de acceso: noviembre 6, 2025, https://www.christenseninstitute.org/graphic/infographic-when-to-use-the-tools-of-cooperation/
56. Notes and Takeaways from The Tools of Cooperation and Change - Rick

Lindquist, fecha de acceso: noviembre 6, 2025, https://www.ricklindquist.com/notes/the-tools-of-cooperation-and-change
57. The Tools of Cooperation and Change - Pickard & Laws Consulting Group, fecha de acceso: noviembre 6, 2025, https://www.pickardlaws.com/myleadership/myfiles/rtdocs/hbr/Tools%20of%20cooperation%20and%20change.%20%20HBR%20Oct06.pdf
58. Tools of Cooperation - Product Science, fecha de acceso: noviembre 6, 2025, https://www.dx.mba/lenses/tools-of-cooperation
59. Custom vs COTS for Inventory Management - Susco Solutions, fecha de acceso: noviembre 6, 2025, https://suscosolutions.com/custom-vs-cots-inventory-management/