10/02/2026

Agentic Commerce GTM

Agentic Commerce GTM for Custom Apparel

Presented By:
Laura Barbosa

Agenda

1. The agentic commerce shift
– Why custom apparel is changing
– From funnels to agent-mediated networks
– What "winning the agentic algorithm" means
(slides: 03, 06,07)

2. How agents turn intent into products (selection)
– Agentic product selection
– Agentic design generation
– From configuration to purchase
– The decision pipeline
– Conversational search as implementation detail
(slides: 09, 10, 11, 12)

3. How we build this safely
– Key risks & de-risking
– Roadmap
(slides: 13, 14)

4. Demo time!
(slide 16)

Custom Apparel case

Our focus company *(what?)*

For this interview, I built on the assumptions:

There is a custom apparel platform

With strong design expertise

Agent-native ready

Custom Apparel Market *(why?)*

Highly complex purchase decisions

Multiple step configuration

Browser first, manual filtering

High cart abandonment

The agentic opportunity *(how?)*

What if users could:

State their goal: "Team shirts for July event, <€40"

Get viable options (product + design) instantly

Review final configurations, not intermediate steps

Off-property agentic commerce

Agents that optimize for discovery and narrowing.

Generative AI adoption

The adoption of generative AI is now mainstream.

in the EU

Roughly one-third (32.7 %) of people aged 16–74 in the European Union reported using generative AI tools in 2025.

in the US

Surveys show ~60 % of adults use AI for information search, with ~74 % of under-30s doing so.

globally

LLM tools like ChatGPT are used by hundreds of millions weekly worldwide (~800 M users).

Customers already ask AI for shopping help. The question is: are they using YOUR agent or someone else's?

The Agentic Commerce Shift

2025: Funnel-Based	2026: Agent-Mediated
User searches "team shirts"	User tells agent: "Sustainable team shirts for July event, <€40"
Browses 200+ products manually	Agent scouts, filters constraints
Applies filters one by one	Agent evaluates against all criteria simultaneously
Compares 10-20 finalists	Agent returns 2-3 ready-to-buy configurations
Adds to cart → checkout	User reviews → approves purchase

Brands compete to be the top-ranked data point in an agent's selection matrix.
The shift is about:

1. "Winning the algorithm, while complementing the funnel"
2. ”Achieving mission-based style"
3. "Aided curation + supervised execution"

Properties to Target

Different agent surfaces optimize for different things, so brand needs change accordingly.

Property	Protocol layer	Use case	Availability
Google suite (Gemini/Search)	UCP	Agents can query variants, availability, constraints	US only
OpenAI / Stipe ecosystem (Chatgpt)	ACP	Enables configuration → cart without manual steps	US only, eligible already if on Etsy/Shopify
Other LLMs, marketplaces, agents	1) TOON 2) MCP	1.data serialization format used *with* major LLMs 2.utilizing defined schemas to externalise tool usage	Implementable

Off-property strategy accelerates the on-property strategy by pre-structuring the data agents need.

On-property agentic commerce

Agents are curators of the customer experience.

Track 1: Agentic Product Selection for Inventory discovery

A brand-owned agent has the goal of maximizing purchasing confidence to increase conversion.

The agent reasons over inventory to find viable products:

Interpret & refine intent

When intent is goal (not a query), an agent: maps intent to what's actually possible, aligns with brand-specific constraints, clarifies edge cases (fit, sizing, quantity, timing)

Act as a stylist + configurator

An agent can: select the right base product, reasons about aesthetics (style, color, silhouette), enforces technical constraints (price, fabric, print area, availability) and produces ready-to-buy configurations

Supervised execution

An agent prepares cart-ready output, user reviews outcome, not steps, confirmation gates protect trust

12

Track 2: Inventory Design Configuration

The design expertise becomes the agent's constraint engine – ensuring every generated design is production-ready

Generate

Explore design space
Slogans, icons, imagery
Layout & placement
Colour & contrast
Multiple variants

→

Validate

Print area constraints
Brand guidelines
Colour contrast rules
Production feasibility
Quality checks

→

Configure

Apply validated design to product
Final pricing
Production-ready specs
Cart preparation

Supervised Execution: Trust Through Transparency

Agents prepare the purchase — users authorize it.

Prepare

The on-property agent assembles a cart-ready configuration:

- validated product + design
- correct pricing and availability
- delivery and timing constraints

Output is deterministic and inspectable.

Review

The user evaluates the outcome, not the process:

- preview of product and design
- price and delivery summary
- ability to make small adjustments

Cognitive load is reduced to a single decision.

Approve

Execution is gated by explicit user consent:

- add to cart
- checkout handoff
- no autonomous spending

Trust is preserved through supervision.

From Intent to Product: The Agent’s Decision Pipeline

Conversational search extracts intent | Agent pipeline executes reasoning

Key Risks in Agentic Commerce

Risk	Impact	De-risk strategy
Hallucinated recommendations	Users are shown invalid or unavailable products	• Hard product fact constraints • Real-time inventory & availability validation
Unproducible designs	Print failures, brand violations, rework	• Automated print & brand constraint checks • Template-first, constrained design generation
Failure to gain user trust	Users hesitate to delegate decisions or abandon flows	• Autonomy stops at commitment (agent prepares, user approves) • Transparent previews, pricing, and delivery reasoning
Fulfilment promise violations	Missed delivery dates, incorrect quantities, SLA breaches	• Agent bound to fulfilment constraints (lead time, MOQ, cutoff dates)

Every agent capability must be inspectable, and reversible.

Key Success Metrics *(V.0)*

Conversion Rate

Agentic design validation pass rate

Target: 80%

Time to Purchase

Time to confidence purchase

Target: 8 min

Design Validation

Conversion rate of agent-mediated sessions

Target: +20%

Phase 1 goal: Prove agents outperform manual browse before touching transactions.

Demo time!

https://merch-builder-app.vercel.app/

Username: demo

Password: agentic1234

MerchForge

Agentic flow with guardrails:

What's agentic

LLM-driven state machine — Both the /create and /discover flows use an LLM to advance a multi-stage conversation. The LLM returns structured JSON with an assistant message *plus* updates that mutate the conversation state (stage, product selection, constraints, etc.). The app then acts on those updates to progress through stages like welcome → product → text → icon → preview → complete.

LLM decides what to extract — In the create flow (lib/agent-llm.ts), the model autonomously decides which fields to populate (product, color, text, icon, size, quantity) and can trigger actions like add_to_cart or remove_icon based on the conversation.

Self-correction retry — In agent-llm.ts:153-158, if the LLM returns invalid JSON, it feeds the bad output back and asks the model to try again — a basic agentic self-repair loop.

LLM-guided ranking — In the discover flow (api/discover/route.ts), the LLM can return a selection object with primaryIds and fallbackIds to reorder results, plus a rationale explaining its choices.

What's NOT agentic

No tool use / function calling — The LLM never invokes tools, calls APIs, or takes autonomous multi-step actions. It's always a single LLM call (with at most one retry) per user message.

No planning or reasoning loops — There's no chain-of-thought, no multi-step plan execution, no branching decision trees. The LLM is essentially a structured-output parser that also generates a friendly reply.

No memory beyond the session — No persistent memory, RAG, or learning across conversations.

Heavy deterministic fallback — The discover flow has a full keyword-based parseConstraints() function (lib/discover.ts:59-130) that extracts constraints with regex. If the LLM fails entirely, the app still works via fallbackResponse(). The LLM augments the deterministic logic rather than replacing it.