Agentive Documentation Assistance Model

From Grunts to Great UX: The Evolution to Proactive and Personalized User Assistance

MICHAEL IANTOSCA
SENIOR DIRECTOR OF KNOWLEDGE PLATFORMS AND ENGINEERING
AVALARA INC.

Since the dawn of humanity, the industry's approach to documentation, user assistance, and technical support has been firmly stuck in a *reactive failure model*. Picture it: a prehistoric human, let's call him Grug, wants directions to the nearest watering hole. What does he do? He consults the ancient documentation—a series of cryptic cave wall etchings. If Grug can't decipher the scratches, he's out of luck. Enter "live technical support": a fellow tribe member grunting and gesturing vaguely, leaving Grug wondering if he's heading to a watering hole or a pack of saber-toothed tigers. He can't even be sure if his chatbot, a pet parrot, is telling him the truth.

This, my friends, is failure-mode assistance that includes documentation, user assistance, costly technical support, and even generative AI chatbots. These provide help only when things go wrong and are typically just as confusing as the problem itself. They force the "user" (Grug) to abandon his task and seek external intervention to get back on track. Failure mode is defined as any interruption to accomplishing users' goals, plain and simple.

But what if prehistoric UX had been proactive? Imagine Grug embarking on his journey and encountering stacked stones or tree markings at key intersections, guiding him naturally along the correct path. There would be no need to trek back to the cave wall or summon his grunting guide—directions would be right there, just in time, intuitive, and obvious. It's the stone-age equivalent of a frictionless, perfectly designed user interface that remains eternally elusive for powerful, multi-purpose, and customizable applications.

Now, fast forward to the year 2025. Surely, we've left Grug's rudimentary systems in the dust, right?

...Right?

Instead, we're still creating tools and interfaces that throw users into the same frustrating loop: cryptic instructions, confusing paths, and a reliance on external help. Modern apps might not involve cave etchings, but how often have you had to stop what you're doing to Google how to complete a task, scroll through forums and FAQs hoping some digital tribe member has posted a workaround, or toss up your hands and open a technical support ticket and more recently, ask a chatbot?

We should strive for *proactive* user assistance, like Grug's trail markers—help that's embedded, self-evident, and meets users where they are rather than making them retrace their steps or grunt for backup. So, until we all embrace proactive methods, Grug might be more relatable than we care to admit. Let's give Grug—and ourselves—a break and design for the journey, not just the mishaps.

Personification versus personalization

Let's face it: what we often call "personalized content" isn't personalized at all—it's a dressed-up generalization. True personalization, the dream of tailoring content to an individual's exact needs, context, and moment of need, is like a unicorn: theoretically possible but rarely, if ever, seen. Instead, we mostly get *personification*, where content is generalized to broad user categories—personas—masquerading as individual customization.

This confusion isn't new. Way (way!) back in the 1980s, while my IBM colleague and I tinkered with cutting-edge multimedia at the dawn of the personal computer, we noticed something about content architecture: it was rooted in static, book-style paradigms, even as object/topic-based models were emerging. It was an exciting time, years before the web brought Mosaic and HTML into the spotlight, but even then, the vision for true personalization was clear. The problem? The technology to bridge the gap wasn't there yet.

Instead of crafting content that anticipated and resolved a user's unique needs, we settled for *persona-based generalization*, wrapped it in a bow, and called it "personalization." But let's be honest: it wasn't personalized, and it still isn't —it was tailored for broad swathes of users who vaguely fit a predefined mold.

Fast-forward to today, and the situation hasn't evolved as much as we'd like to think. Yes, we've made strides—object models, advanced analytics, and AI are tools we dreamed of back then. But for most systems, personalization remains a façade, relying on demographic buckets, past behaviors, and educated guesses.

What we need—and what we were only imagining in the 1980s—is proactive, user-specific assistance that doesn't just react to failure but anticipates needs in real-time. Interestingly, the Expert Systems we explored back then, with their rules-based logic, *symbolic reasoning*, and knowledge representation (programmed in LISP, Prolog, and Smalltalk), share striking similarities to today's generative AI. We're finally circling back to a realization that true personalization, like the dreams of early AI pioneers, is finally within reach.

The elements to achieve proactive, personalized assistive

Thirty-five-plus years is a long time to wait for a true paradigm shift, but all the technology to make it practical at scale finally exists.

We're come a long way with content and knowledge management since that era.

- Most of the content industry advanced from writing monolithic chapter and book models to object, component, and topic-based content models.
- We made content *intelligent* for machine processing with structured formats such as XML and DITA. We created content that is device and channel independent. Intelligent content is self-describing and organized systematically so machines can quickly identify and process it, making it reusable, adaptable, and discoverable while enabling automated production workflows and dynamic delivery.
- A relatively new class of user behavior analytic technology called digital experience analytics tools has become commonplace. Digital experience analytics tools, such as

session replay tools, user journey analytics, experience analytics tools, customer journey mapping tools, and behavioral analytics tools, are designed to track and analyze how users interact with software applications. These platforms record user sessions, visualize their paths and actions, collect engagement data, and provide insights into user behavior and friction points, helping optimize usability and improve the overall user experience. Digital experience analytics tools provide real-time monitoring and analytics for in-app user activity and feature adoption, along with APIs for retrieving analytics, usage data, and triggering in-app messages or guides.

- Retrieval-augmented generative (RAG) AI models provide dynamic assistance, such as chatbots and agentive assist, but most vector-based models remain probabilistic and lack the certainty needed for agentic AI. Newer, knowledge graph-based RAG models provide more accurate, flexible, reliable, and explainable AI based on symbolic reasoning and knowledge representation. Graph RAG models for content, such as our Document Object Graph RAG model, provide contextual retrieval to power advanced AI models.
- Agentive AI applications are emerging. Agentive AI acts autonomously on behalf of users or machines to accomplish specific tasks. Unlike tools that require constant input, agentive AI proactively takes the initiative, learning from preferences or data to perform actions with minimal user direction.

These are precisely the combination of modern technologies needed to cross the gulf between reactive, failure-mode user assistance. With these, we can meet the individual user with the right user assistance at the precise moment and context in which they need it, even proactively before users know they need it. That is a genuine paradigm shift.

Extending our DOM Graph RAG Model to provide real-time personalized user assistance

Imagine this: You're using an app, and instead of stumbling through features or searching FAQs, it feels like the app just "gets" you. That's the vision behind integrating digital experience analytics tools with a retrieval-augmented AI RAG framework—a system designed to deliver real-time, personalized user assistance that evolves as you interact.

Here's how this next-gen setup might work: the digital experience analytics tool tracks your in-app activity, feeding insights into an RDF-based graph that models user behavior and relationships. This graph, paired with retrieval-augmented AI, creates context-aware guidance delivered right where you need it. The result? Frictionless, intuitive user experiences.

What Is This System?

This proposed system combines our DOM Graph RAG model with a digital experience analytics tool's analytics and in-app guidance features to:

1. Track user interactions within the application in real time.
2. Use an RDF graph to model and store user behavior, insights, and inferred relationships.
3. Provide personalized, context-aware assistance directly within the application via the digital experience analytics tool.

4. Continuously improve its recommendations through a feedback loop that refines the RDF graph with each interaction.

Core Components:

1. **DOM Graph RAG Model:** Maintains the structured representation of document components and retrieval-based insights. See https://medium.com/@nc_mike/document-object-model-graph-rag-af8ae452b0b6 for a detailed description of the DOM Graph RAG model we previously published and an open and extensible model.
2. **Digital experience analytics tool Integration:** Tracks feature usage, user flows, and behavioral data, enabling real-time monitoring and guidance.
3. **RDF Graph:** Stores semantic relationships and inferred insights, powering the reasoning engine.
4. **Inference Engine:** Applies logic or machine learning to derive actionable insights from user data.
5. **Feedback Loop:** Updates the RDF graph and improves guidance based on new user interactions.

Why build this system?

Modern applications often overwhelm users with features and complexity, leading to frustration, underutilized capabilities, and poor user retention. This system addresses these challenges by:

1. **Personalizing User Experience:** Provides tailored guidance based on individual behavior, making applications more intuitive.
2. **Improving Feature Adoption:** Highlights underutilized features and promotes their usage to maximize user value.
3. **Enhancing Efficiency:** Identifies common workflows and suggests optimizations, reducing time-to-value for users.
4. **Creating a Data Flywheel:** Continuously refine the system's understanding of user behavior by applying agentive inferencing to expand the graph, leading to smarter assistance that continually learns and improves.

How to build this system

1. Define Objectives for Assistance

Begin by outlining the types of assistance you want to provide. For example:

- **Guidance:** Offer in-app tooltips or walkthroughs for complex workflows.
- **Reminders:** Notify users about overlooked or underutilized features.
- **Interventions:** Help users with difficulties, such as repeated errors.

- **Assistance:** Launch do-with or do-for agents to perform the task or guide the use through the user interface step-by-step

2. Structure the RDF Graph

The RDF graph is the central knowledge base, capturing entities, relationships, and inferred insights. Define the schema to represent:

- **Entities:** Users, application features, user sessions, and insights.
- **Relationships:** Examples include:
    - User123 interactsWith FeatureX
    - FeatureY isUnderutilized
    - User123 needsAssistance FeatureZ

To store and query your RDF data, use a graph database with an inference engine, such as GraphDB.

3. Integrate the digital experience analytics tool for Real-Time User Monitoring

The digital experience analytics tool provides the ability to track user interactions and deliver in-app guidance:

1. **Install the digital experience analytics tool SDK:** Embed the digital experience analytics tool's SDK into your application to start tracking user activity.
2. **Define Events:** Identify key user interactions to monitor (e.g., feature usage, button clicks, errors).
3. **Set Up Data Capture:** Configure the digital experience analytics tool to send this interaction data to your middleware in real time via APIs.

Example of the digital experience analytics tool Event Payload:


```
{
  "userId": "123",
  "event": "feature_click",
  "feature": "dashboard",
  "timestamp": "2024-12-15T10:00:00Z"
}
```


4. Develop the Middleware Agent

The middleware agent acts as the processing hub, connecting the digital experience analytics tool with the RDF graph:

1. **Technologies:** Build the middleware using Python libraries such as FastAPI and Flask or the Node.js Express.js framework.
2. **Functions:**
    - **Ingest Events:** Receive event data from the digital experience analytics tool.
    - **Query the RDF Graph:** Use SPARQL to retrieve context about the user and their interactions.
    - **Process Data:** Apply inference rules or machine learning models to derive insights.
    - **Update the RDF Graph:** Insert new relationships or insights.

Example SPARQL Query:

```
SELECT ?feature WHERE {
  ?user ex:interactsWith ?feature .
  FILTER (?user = ex:User123)
}
```

Example SPARQL Insert:

```
INSERT DATA {
  ex:User123 ex:needsAssistance ex:FeatureZ .
}
```

5. Build the Inference Engine

The inference engine processes user data to generate insights. Choose from:

- **Logic-Based Rules:** For example, if a user hasn't accessed a key feature in a specific timeframe, infer that it's underutilized.
- **Machine Learning Models:** Train models on historical data to predict user frustration, feature preferences, or subsequent actions.

6. Deliver In-App Guidance with the digital experience analytics tool

Use the digital experience analytics tool's in-app messaging tools to act on insights derived from the RDF graph:

1. **Tooltips:** Provide immediate, context-aware guidance for underutilized features.
2. **Walkthroughs:** Offer step-by-step guidance for complex workflows.
3. **Notifications:** Alert users about new features or tips tailored to their behavior.

For example, if the RDF graph indicates that User123 needsAssistance FeatureZ, trigger a tooltip in the digital experience analytics tool:

"Need help with FeatureZ? Click here for a quick guide!"

7. Establish a Continuous Feedback Loop

The system should evolve by:

1. Capturing and analyzing new user interactions.
2. Updating the RDF graph with enriched insights.
3. Refining inference rules or retraining ML models based on new data.
4. Measuring the effectiveness of guidance provided via the digital experience analytics tool and iterating accordingly.

A sample workflow

1. A user interacts with "FeatureX" but does not complete the intended workflow (captured by the digital experience analytics tool).
2. The middleware receives the event and queries the RDF graph for context about the user and feature.
3. Based on predefined rules, the middleware infers that the user may need guidance.
4. The RDF graph is updated: INSERT { User123 needsAssistance FeatureX }.
5. The digital experience analytics tool delivers a contextual tooltip: "Having trouble with FeatureX? Here's how to complete the process."
6. User behavior is monitored to assess whether the guidance resolves the issue, and this outcome informs future iterations.

The future of assistance: A leap from Grug's cave to real-time, proactive support

Let's take a moment to reconsider Grug's predicament, our prehistoric traveler struggling with cryptic cave etchings and his pet parrot chatbot. Imagine if Grug had encountered static instructions and a system that guided him along his journey in real-time, anticipating his needs before he even recognized them. That's the promise we're on the verge of realizing by integrating the digital experience analytics tool into a reliable generative AI model such as our DOM Graph RAG model.

Just as Grug longed for an obvious and intuitive path, today's users expect nothing less than assistance seamlessly woven into their experience—help that adapts in real-time, aligns with their unique context, and is delivered at the right moment. Gone are the days when users had to retrace their steps or search for guidance. With the system we've outlined—where the digital experience analytics tool tracks interactions, the RDF graph stores context, and AI-driven insights provide

personalized assistance—we're finally bridging the gap from reactive, failure-mode support to proactive, personalized help.

This is the true paradigm shift we've been waiting for since the days of Grug. By combining semantic knowledge representation, real-time analytics, and AI-powered inference, we can craft a user experience that doesn't just respond to problems—it anticipates them. No more hunting through cave walls or waiting for a grunting guide to appear. Instead, assistance is intuitive, relevant, and designed to keep users on track, making the path forward as clear as Grug's trail markers—just more sophisticated and infinitely more precise.

This approach isn't just a technological leap; it's a *philosophical* one. It's the moment we finally leave behind the reactive, failure-driven assistance of the past and embrace an era where help isn't just available—it's proactively integrated into the user experience, anticipating needs before users even realize them. And that's a true evolution—from the caveman's struggle to a world where assistance is always there when needed.