Agentic Workflows for Business Analysts

"The goal isn't to remove the analyst from the process. The goal is to remove the bottleneck that prevents the analyst from applying their expertise where it matters most."

Steven Duchene | Business Analyst / Product Owner | August 2026

Abstract: Business Analysts spend the majority of their time on research, documentation, and context-gathering — not on the decision-making that creates value. This paper demonstrates how purpose-built agentic workflows can compress multi-hour analysis tasks into minutes while maintaining quality and traceability. Drawing from six real-world implementations in a large-scale unemployment insurance modernization project, we show measured time reductions of 70–90% on documentation, investigation, and data analysis tasks — without sacrificing accuracy or the analyst's judgment role.

Contents

The BA Bottleneck

In a typical enterprise software environment, Business Analysts are the translators between what clients need and what developers build. This role requires deep technical context, institutional knowledge, and the ability to produce precise documentation under time pressure.

The problem isn't that BAs lack skill — it's that the skill gets buried under volume. A senior BA on a modernization project might juggle 40–60 active OPCs (change orders) simultaneously across multiple clients. Each ticket requires:

- Researching the current system behavior across 4+ architectural layers
- Documenting findings in a specific format the client and developers can consume
- Writing specifications that must be both technically precise and client-readable
- Investigating incidents by tracing data through code, configuration, and database
- Generating test paths that cover all edge cases without ambiguity

Industry research quantifies this burden. IIBA's 2026 practitioner research reports that approximately **35% of enterprise project effort** is consumed by knowledge transfer activities alone — business workshops, alignment sessions, architectural walkthroughs, and developers inferring intent from code.[1] A separate survey of 94 enterprise practitioners found that **76.5% cite insufficient time and resources** as the primary obstacle to maintaining effective documentation.[2]

Combined with documentation production and system research, practitioners consistently report that 70–80% of their working time goes to information-gathering and deliverable-production rather than the judgment-intensive work that creates primary value. Our own observations from a 10-month implementation on a large-scale modernization project confirm this ratio:

Activity	Est. % of Time	Value Created
Reading code and tracing logic across architectural layers	25–30%	Context (enables decisions)
Writing documentation (requirements, specifications, rules)	25–30%	Deliverable (formulaic structure, high effort)
Querying databases to verify behavior and configuration	10–15%	Evidence (confirms or refutes assumptions)

Formatting, organizing, publishing output	10%	Presentation (necessary but low-skill)
Design decisions, client communication, judgment calls	20–25%	**Primary value — irreplaceable**

The pattern is clear: 75–80% of BA time goes to activities that are *necessary* but *formulaic* — they follow patterns, they require attention to detail, but they don't require human judgment at every step. This is the bottleneck: the high-value 20–25% is constrained by the bandwidth consumed by the other 75–80%.

IIBA's 2026 Global State of Business Analysis Report reinforces this from the AI adoption side: the most common uses of AI among BAs are drafting and formatting (28%), documentation and communication (25%), and idea generation or problem-solving (22%).[3] The profession is already signaling where the leverage is. Agentic workflows are the structured, reliable implementation of that signal.

> **Sources:**
> [1] Civelek, N. "Why Enterprise Software Documentation Needs Clear Ownership." IIBA Analyst Catalyst Blog, February 2026.
> [2] Civelek, N. "Survey Results: Could Documentation Be the Missing Link Between Strategy and Execution?" LinkedIn, 2025.
> [3] IIBA. *2026 Global State of Business Analysis Report*. 2,120 respondents across 122 countries.

What is an Agentic Workflow?

An agentic workflow is not a chatbot. It's not autocomplete. It's a structured, multi-step process where an AI agent executes a defined procedure — reading files, querying databases, tracing code paths, assembling output — with human checkpoints at decision points.

The distinction matters:

Simple AI Chat	Agentic Workflow
User asks a question, gets an answer	User triggers a process, agent executes a pipeline
No persistent state	Maintains context across steps, builds on prior work

User drives every action	Agent drives execution, user provides judgment at gates
Generic output	Output matches organizational format, tone, and conventions
One-shot interaction	Iterative: draft → refine → deliver

Think of it as the difference between asking someone a question and delegating a task. You delegate when you trust the process, the output format is known, and you retain the authority to accept or modify the result.

Design Principles

Effective agentic workflows share several design characteristics that make them reliable enough for professional use:

1. Hard Gates

Critical prerequisites that must be satisfied before the agent proceeds. In our Business Rules Extraction workflow, the agent cannot begin extraction without first completing a domain analysis document. This prevents the most common failure mode: producing superficially correct output from incomplete information.

2. Interview Pattern

For tasks requiring human context (what client, what scope, what constraints), the agent asks structured questions one at a time before executing. This front-loads the human input where it has the highest leverage — scoping — and then lets the agent run autonomously through execution.

3. Style Calibration

Production-quality output must match organizational voice. Our ARD (Agile Requirements Document) workflow reads the analyst's previous documents to extract writing patterns — trigger structure, detail level, phrasing conventions — and applies them to every generated draft. The output reads like the analyst wrote it because it learned from their prior work.

4. Layered Output

Most deliverables serve multiple audiences. A Business Rules document has a "Client Readable" section (no code, no file paths) and a "Developer Details" section (exact line numbers, method signatures, branching logic). The agent produces both from the same analysis pass.

5. Verification Loops

The agent doesn't guess. When static analysis is ambiguous, it executes queries against the database to confirm actual configured values. When a sysparam's effect depends on client configuration, it reports the verified value — not the code default.

Case Studies

CASE STUDY 1

OPC Lifecycle Timeline Analysis

The Problem: Leadership needed to understand how long change orders take from creation to UAT approval — and how much of that time is spent waiting on the client vs. GSI internal work. The data lived in 835 individual ticket records with status change history buried in unstructured notes.

The Manual Process: Export ticket data to spreadsheets. Manually open each ticket to find status change dates. Calculate durations in Excel. Build charts. Write observations. Estimated effort: 2–3 full days for a meaningful sample.

The Agentic Process:

1. User provides OPC list (835 numbers pasted)
2. Agent batch-queries all 835 tickets for metadata (20 minutes automated)
3. Agent samples 62 completed OPCs, pulls their full notes, and parses status change timestamps using regex patterns
4. Agent calculates phase durations, client vs. GSI time splits, and distribution statistics
5. Agent generates a professional HTML report with stat cards, bar charts, tables, and observations

Time: 15 minutes of wall-clock time (mostly API calls). Zero manual data entry. Zero spreadsheet work.

> **92%** reduction in effort — from ~20 hours to ~15 minutes of analyst attention

Output: Data-driven report showing median lifecycle of 195 days, with 60% of elapsed time in client-facing statuses. Immediately actionable for leadership conversations about process improvement.

CASE STUDY 2

Business Rules Extraction

The Problem: Documenting the exact business rules for a VOS page requires reading across 6–10 source files (UI markup, code-behind, managers, data layer, stored procedures, configuration), tracing the execution flow, and producing both client-readable and developer-facing documentation.

The Manual Process: Open each file. Read and understand the logic. Cross-reference configuration values. Write up rules in the organizational format. Include revision history. Estimated effort: 1–3 days depending on page complexity.

The Agentic Process:

1. Agent verifies a domain analysis exists (hard gate — forces complete reading before summarization)
2. Agent reads all layers: UI, code-behind, manager, DL, stored procedures, configuration
3. Agent traces program flow and identifies discrete business rules
4. Agent queries the database for actual sysparam values and help text
5. Agent produces a styled HTML document with: page defaults, purpose, access/privileges, page layout with grid mockups, grouped business rules (client-readable + developer details), sysparam reference, data source queries, and revision history

> **75%** reduction in effort — from 1–3 days to 2–6 hours (including review and refinement)

Key Advantage: Completeness. The agent reads every line of every related file. Human analysts, under time pressure, inevitably skim. The agent's output includes edge cases and inactive code paths that manual extraction would miss.

CASE STUDY 3

ARD Generation (Agile Requirements Documents)

The Problem: ARDs follow a strict organizational template with specific section requirements. Each one requires understanding the current system state, the proposed change, affected areas, and a detailed "Required Changes" section with trigger-based formatting. Writing a quality ARD takes 1–2 hours even when the analyst knows the requirements well.

The Agentic Process:

1. Agent reads the analyst's style reference (calibrated from 3–5 prior ARDs)
2. Agent pulls OPC details and all notes from the ticketing API
3. Agent asks targeted questions: BRD requirement number, discussed with other states?
4. Agent researches the affected page in the codebase to identify the URL, current behavior, and technical details
5. Agent generates the complete ARD matching the analyst's personal writing style

Before (Manual)

- Pull up OPC notes
- Research the page/code
- Open Word template
- Write each section
- Format consistently
- ~60–120 minutes

After (Agent-Assisted)

- Say "ARD for [OPC]"
- Answer 2 questions
- Review generated draft
- Minor adjustments if any
-
- ~5–10 minutes

Key Advantage: Style calibration means the output doesn't read like AI — it reads like the analyst. Trigger structures, phrasing conventions, and level of detail all match the analyst's established patterns.

CASE STUDY 4

Incident Investigation

The Problem: When a client reports something isn't working, the BA must determine whether it's a true code defect, a configuration gap, user error, or expected behavior. This requires querying data, tracing code paths, checking configuration, and often reviewing related tickets — before a classification can be made.

The Manual Process: Read the OPC notes to understand the symptom. Log into the database to query the affected records. Open source files to trace the code path. Check sysparams. Check for related OPCs. Write up findings. Estimated effort: 1–4 hours depending on complexity.

The Agentic Process:

1. Agent pulls OPC details, all notes, and linked records automatically
2. Agent extracts the reported symptom, expected behavior, and information gaps
3. Agent classifies the symptom type and follows the appropriate investigation path
4. Agent queries the database to verify data state
5. Agent traces the code path for the scenario described
6. Agent produces a structured triage report with classification, confidence level, reasoning, and recommended next steps

> **70%** reduction in investigation time — structured output that cites specific evidence for every determination

Key Advantage: The structured triage report forces rigorous classification. Every determination must cite evidence — no "I think it's working as expected" without showing what was checked. This eliminates the common failure mode of premature classification.

CASE STUDY 5

Page Mockup Generation

The Problem: When designing a new page or modification, BAs need to produce visual mockups that look like the actual product — matching the real UI controls, styling, layout patterns, and branding. Traditional tools (Figma, Balsamiq, PowerPoint) require separate skills, separate tool licenses, and produce output that often looks nothing like the finished product.

The Agentic Process:

1. Agent conducts a brief interview: page name, user type, sections, fields per section
2. Agent generates a static HTML mockup using extracted CSS from the actual product (real class names, real styling, real header screenshots)
3. Agent includes realistic sample data, proper control types, entity header bars, and jump links
4. Agent supports iterative refinement: "add field X to section Y", "move section Z above W"

Before (Manual Mockup)

- Open design tool
- Recreate product styling
- Add fields manually
- Screenshot or export
- Paste into document
- ~1–2 hours per page

After (Agent-Generated)

- Describe sections and fields
- Agent generates full HTML page
- Open in browser — pixel-accurate to product
- Iterate with natural language commands
-
- ~10–15 minutes per page

Key Advantage: The mockup uses the actual product's CSS. When stakeholders see it, they're looking at something visually identical to the real system — not an approximation. This eliminates the "it doesn't look like that" feedback loop.

CASE STUDY 6

Test Path Generation

The Problem: Every change order requires a test path before QA can validate the work. A quality test path needs: prerequisites, data setup instructions, numbered steps, explicit pass/fail criteria, and edge case coverage. The BA must understand what the code actually does (not just what it should do) to write a complete test.

The Manual Process: Read the ARD and dev implementation notes. Trace the code to understand trigger conditions. Check what test data exists. Write step-by-step instructions. Think through edge cases. Estimated effort: 30 minutes for simple changes, 2+ hours for complex ones.

The Agentic Process:

1. Agent pulls OPC details, notes, and implementation specifics
2. Agent traces the code to identify all trigger conditions and prerequisites
3. Agent queries the test database to verify available test data
4. Agent generates complete test path: objective, prerequisites, data setup, step-by-step scenarios with explicit AND/OR pass/fail criteria, and edge cases
5. Agent validates the test path against the code to ensure all conditions are covered

> **80%** reduction in effort — comprehensive test paths generated in minutes instead of hours

Key Advantage: The agent reads the code to determine exact trigger conditions — it doesn't rely on potentially incomplete or outdated documentation. Test paths cover conditions the BA might not think to check because the agent traces every branch.

Quantified Results

The six case studies above are illustrative, not exhaustive. The same design principles have been applied to over a dozen additional workflows, including:

- **CCB Decision Support** — structured impact/risk/benefit analysis for Change Control Board review
- **Team Operational Dashboards** — real-time queue visibility with next-action recommendations for every team member
- **Report Requirements Analysis** — searching existing reports before building new, preventing duplicate effort
- **System Alert Tracing** — mapping notification templates through multi-layer address resolution and data swap logic
- **Knowledge Base Maintenance** — discoveries documented in reusable, indexed format that survives team turnover
- **Demo Script Generation** — interview-driven, timed presenter scripts with navigation-accurate paths

Each follows the same pattern: structured process, human checkpoints at judgment points, verified output, and organizational format compliance.

Across all workflow categories implemented over a 10-month period:

Workflow	Manual Effort	Agent-Assisted	Reduction
OPC Lifecycle Analysis	~20 hours	~15 minutes	99%
Business Rules Extraction	1–3 days	2–6 hours	75%
ARD Writing	60–120 minutes	5–10 minutes	90%
Incident Investigation	1–4 hours	15–30 minutes	70–80%
Page Mockups	1–2 hours	10–15 minutes	85%
Test Path Generation	30 min–2 hours	5–15 minutes	80%

> **Cumulative Impact:** For a BA managing 40–60 concurrent OPCs across multiple clients, these workflows collectively recover approximately 15–20 hours per week that can be redirected to high-value activities: design discussions, client negotiations, risk assessment, and mentoring.

Quality Indicators

Speed without quality is counterproductive. Key quality observations from production use:

- **Completeness improved:** Agent-generated Business Rules documents consistently capture 15–20% more rules than manual extraction because the agent reads every line of every file — no skimming under time pressure.
- **Format consistency:** 100% adherence to organizational templates. No forgotten sections, no inconsistent heading styles, no missing table-of-contents entries.
- **Traceability:** Developer Details sections include exact file paths and line numbers. Every claim about system behavior can be verified.
- **Error reduction:** Fewer instances of "I thought it worked that way" because the agent verifies against actual database configuration rather than relying on memory.

What Agents Cannot Replace

This paper deliberately focuses on tasks where agentic workflows excel. It's equally important to identify where they don't — and shouldn't — replace human judgment:

Task	Why It Remains Human
Client relationship management	Trust, empathy, and political awareness can't be automated
Design decisions	Choosing between valid approaches requires product vision and taste
Priority negotiation	"What should we do first?" is a values question, not a data question
Risk assessment	Understanding what could go wrong in production requires experience and intuition
Ambiguity resolution	When requirements contradict, the BA decides — not the agent

Cross-team communication	Translating between stakeholder perspectives requires social intelligence

The agent handles the *what is* (current state, data, code behavior). The analyst handles the *what should be* (requirements, priorities, tradeoffs). This division of labor is the design point — not a limitation.

Implementation Guidance

For teams considering agentic workflows, the following lessons emerged from this implementation:

Start with Your Biggest Time Sink

Identify the task you do most frequently that follows a repeatable pattern. For us, it was Business Rules Extraction — high-value output, clear structure, but extremely time-consuming because it requires reading across many files. That became the first workflow.

Build Incrementally

No workflow was designed perfectly on the first attempt. The process is: build a basic version → use it → notice what's missing or wrong → refine. Our Business Rules workflow went through 8+ revisions before reaching its current form. Each revision was prompted by a real situation where the output fell short.

Encode Your Standards, Not Just Your Tasks

The most impactful design decision was encoding organizational conventions: document structure, naming patterns, tone expectations, output format. This turns every execution into a standards-compliant deliverable without the analyst thinking about formatting.

Hard Gates Prevent Garbage Output

The single most important reliability technique is the hard gate: a prerequisite that must be satisfied before the agent proceeds. Without our "domain doc must exist" gate, the Business Rules workflow produced plausible but incomplete output 40% of the time. With the gate, that dropped to near zero.

Calibrate to the Person, Not Just the Process

Style calibration — reading an analyst's prior work to match their voice — transforms output quality from "obviously AI-generated" to "reads like I wrote it." This is critical for credibility: stakeholders should not be able to distinguish agent-assisted output from fully manual work.

Verify Against the Source of Truth

Any workflow that makes claims about system behavior must verify those claims against the database or codebase. An agent that says "the sysparam is set to True" without querying the actual database is guessing — and guessing erodes trust.

Conclusion

Agentic workflows don't replace Business Analysts. They replace the hours of mechanical work that prevent Business Analysts from doing what they're actually good at.

The evidence from this implementation is clear: a structured agent operating within defined guardrails can execute the research, documentation, and formatting portions of BA work at a fraction of the time cost — while maintaining or improving quality. The analyst's role shifts from executor to director: scoping the work, making judgment calls, validating output, and communicating results.

For organizations running large-scale enterprise software projects, this shift isn't optional. The volume of change orders, the complexity of multi-client systems, and the speed expectations of modern delivery all exceed what manual processes can sustain. The question isn't whether to adopt agentic workflows — it's how quickly you can identify your highest-leverage opportunities and start building.

> *"The goal isn't to remove the analyst from the process. The goal is to remove the bottleneck that prevents the analyst from applying their expertise where it matters most."*