OneReach.ai

Conversational Design Patterns

When working with conversational AI, before you touch any technology you need to envision the experience you want to create.

At OneReach.ai, we've studied numerous highly-rated experiences over the years and have noted scores of patterns that lead to good experiences. Keeping these patterns in mind as you create a vision for your experiences—and build the framework of the ecosystem that will bring those experiences to life—can help create a paradigm for accelerated service.

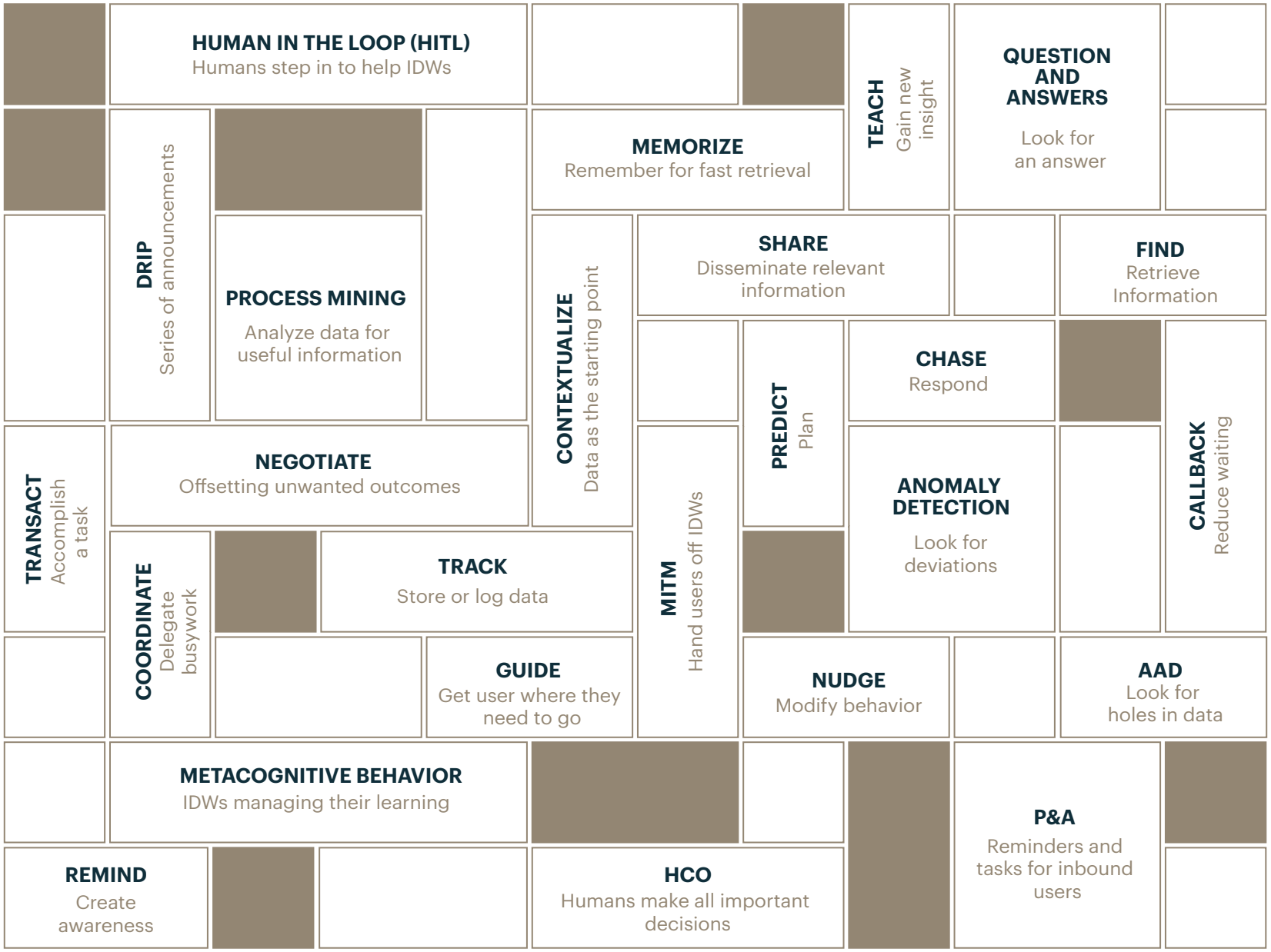
Not only can the sequencing of these patterns efficiently automate tasks with certainty, but these are also the kinds of optimized experiences that will build meaningful relationships between humans and machines. They should be designed as something ongoing that will evolve — not as a series of disconnected transactions, but as contextual relationships. Thinking beyond transactional relationships to contextual relationships creates the opportunity to change the presence of software and machines in our lives so that technology can be always present but never invasive.

This work often begins with in-depth journey maps created by Lead Experience Architects and Design Strategists. These journey maps are used to define the experiences that make up your ecosystem, and they come to life as key patterns are identified and later fleshed out through production design. The real value comes from sequencing technologies to create experiences that are improvements on current workflows and provide better-than-human experiences.

Patterns in Action

To help contextualize these patterns, recognize that many of them are frequently sequenced around GPS technology.

Imagine this scenario: you've got some extra time as you're driving through an unfamiliar area to a meeting, so you ask your smartphone if there's a coffee shop nearby. This is the QUESTION & ANSWER pattern activated. The answer might come back in the form of pinpoints on a map, using the CONTEXTUALIZE pattern to find a coffee shop closest to you. The system can then GUIDE you, navigating you to the coffee shop. It can also use the PREDICT pattern to identify a traffic accident up ahead that could cause a delay. It can NUDGE you to let you know that you might want to take a side street. This is a multimodal journey that incorporates voice, text, and graphical interface, and it can evolve to include other patterns, like TRANSACT — perhaps letting you order and pay for your coffee ahead of time, so you can be sure to make your meeting on time.


HUMAN IN THE LOOP (HITL)
Humans step in to help IDWs
TEACH
Gain new insight
QUESTION AND ANSWERS
Look for an answer
MEMORIZE
Remember for fast retrieval
DRIP
Series of announcements
PROCESS MINING
Analyze data for useful information
CONTEXTUALIZE
Data as the starting point
SHARE
Disseminate relevant information
FIND
Retrieve Information
PREDICT
Plan
CHASE
Respond
CALLBACK
Reduce waiting
TRANSACT
Accomplish a task
NEGOTIATE
Offsetting unwanted outcomes
MITM
Hand users off IDWs
ANOMALY DETECTION
Look for deviations
COORDINATE
Delegate busywork
TRACK
Store or log data
GUIDE
Get user where they need to go
NUDGE
Modify behavior
AAD
Look for holes in data
METACOGNITIVE BEHAVIOR
IDWs managing their learning
P&A
Reminders and tasks for inbound users
REMIND
Create awareness
HCO
Humans make all important decisions

Key Patterns for Lead Experience Architects and Design Strategists to Keep in Their Back Pocket

- **QUESTION AND ANSWERS (Q&A).** Users ask the machine a question and it looks for the answer using natural language understanding (NLU) and a knowledge base designed for providing answers. The Q&A pattern highlights one area where conversation isn't always the best solution: browsing. Q&A is a way to help resolve a question that the user has, but depending on your use case, giving them the option to browse data, products, or a FAQ page might be more useful.
  Adding a graphical UI in addition to the conversational UI can be a way to solve this. For example, if someone asks about what types of services are available, the IDW can point them to the page on the website that lists all of the services. In ecosystems built for hyperautomation, the IDW can also turn to a human-in-the-loop if it doesn't have an answer.
- **FIND.** Users ask the machine to look up information based on certain queries. The machine queries an API and gets a set of results that it can show the user. This might feedback into known answers for Q&A or help with transactions or establish the user's identity. Even though Q&A and FIND seem similar, they are very different patterns. FIND is a good pattern to employ if Q&A fails to help. In the context of FIND, when a machine's training doesn't allow it to answer the question, it can search external sources, usually via API.
- **CHASE.** CHASE is more aggressive than simple reminders; flows built for this will activate continuously until a certain criterion is met. For example, a proactive pattern would hunt down an answer to a particular question. If a user doesn't provide it, the machine will move on to another user—or else continue to repeat the query until it gets its answer. Often, successful resolution will involve escalation.
- **NUDGE.** NUDGE is a soft push toward a desired outcome, but in a manner that's less intrusive than CHASE. It's designed to provide extra information in a structured way that will either subconsciously motivate users to take a particular action or more distinctly prompt them to consciously make the intended choice. A tangible example would be painted lines on a road that clearly delineate cyclists from drivers.
- **DRIP.** DRIP is a series of announcements; it can be used for reporting and making enhancements that don't require immediate feedback. It's a bit like CHASE without the knowledge base, and providing future context. For example, DRIP would offer: "Don't forget, you have an appointment on Monday at 3:30pm," whereas CHASE would offer: "Please confirm your appointment on Monday at 3:30pm by replying Y or N." DRIP often represents a content journey, one that's presented in a predetermined sequence. For example, a DRIP might have a series of five deliberately spaced out messages that go out to first-time customers as part of an on-boarding experience with a new product.
- **MEMORIZE.** This pattern can also establish conversational patterns at-large across multiple users. MEMORIZE can be used to understand the common conversations and questions that users engage in. Flows built for memorization will store information so that it can be used for reporting, making enhancements to the knowledge base, and providing future context. When designing your conversations, you should make sure that you set them up to store data so that you can utilize MEMORIZE. Ultimately, this context helps to build relationships rather than just one-off transactions.
- **REMIND.** REMIND is a proactive pattern that gives users information at a particular time and in a specific way in order to take action. This could be for an upcoming appointment or to establish a new habit. In order to succeed with this very common pattern, send out reminders over whatever channel your customer prefers. In fact, this is another chance to employ the MEMORIZE pattern—noting preferred channels for engagement. To use REMIND successfully, design around the reminder and create a conversational experience that goes beyond the first obvious step.
- **TRACK.** This is similar to MEMORIZE but isn't necessarily used for long-term memorization. For example, the machine might memorize how many times a user goes from point A to B, tracking that everything between point A to B is being logged and used. Flows that track are keeping in mind a "current state"—as well as all the prior states that led to that point.
- **CALLBACK.** Another proactive pattern, CALLBACK is focused on resuming a prior activity. This is geared toward pausing an activity and setting a follow-up — an interval that could be dependent on a certain amount of time lapsing, or on the emergence of a new piece of data.

- **CONTEXTUALIZE.** With CONTEXTUALIZE, the machine is trying to extract context from the conversation using stored data as a starting point. It will query its contextual storage and try to "continue" from that context. Examples could include using time of day, location, the task at hand, or a prior conversation or message in order to establish context. The CONTEXTUALIZE pattern is beneficial in that it allows the machine to use context to improve the conversational experience by cutting out the need for starting from the beginning—such as asking questions like "Are you a customer? When was your most recent purchase?". This pattern goes hand in hand with the MEMORIZE pattern.

- **GUIDE.** With this pattern, the machine literally guides a user from point A to point B, such as in a scripted conversation or a sequence of questions. GUIDE could also help with a particular sequence over time, like checking in each day to help users stay on track with a specific goal. Flows built to guide keep in mind progression and sequence with the milestones or the ultimate outcome they're meant to achieve. GUIDE is also a critical pattern for the concierge skill, which greets users.
  Concierge evaluates a user's needs using patterns like CONTEXTUALIZE and Q&A; then, it uses GUIDE to connect users to the other skills in the ecosystem that will help them achieve their goals. Without GUIDE, a user would be left guessing what an IDW might be capable of, asking questions that wouldn't help the system differentiate what they're *really* asking for. Interactions that follow the user's lead can be complex and difficult to build, which can lead to a common mistake with conversational design: overpromising or letting users expect that your machine can do more than it actually can. Set expectations for your user by guiding them—rather than imposing on them to guide the machine.

- **TRANSACT.** This pattern helps users accomplish a particular task. There's a goal in mind and a desired outcome; common examples would be scheduling an appointment or ordering a product. TRANSACT can also be used for minor changes to a setting or to add a new communication preference. Flows will have a structured script that needs particular information to complete the task, with task completion being the primary measure.

- **NEGOTIATE.** If someone asks an IDW if you can check into a hotel room early and it replies that check-in time is 4 pm, they might be inclined to call and try and persuade someone to bend the rules their way. Trying to persuading a machine is futile, but you can prevent that phone call by building negotiation into the process. In this case, the IDW can negotiate by saying something like, "If our regular check-in time of 4pm doesn't work for you, let me see if I can try and work something out and get back to you." This gives the user the impression that calling would be futile, so they'll wait to hear back from the IDW—likely employing HITL to get an answer.

- **PROMISES AND ASSIGNMENTS (P&A).** This pattern was inspired by a concept that developers often use in JavaScript that has an asynchronous component to it: someone either promises to take care of a certain task the next time they login, or assigns someone else to take care of a task when they next login. In most scenarios people tend to think of interactions with machines as being either inbound (someone is calling you) or outbound (you reach out or are responding to someone). PROMISES AND ASSIGNMENTS represents a third category, the real-life equivalent of which would be me telling you, "Hey, next time you talk to Teddy, remind him he owes me $100." Within an ecosystem for hyperautomating, this takes the form of a queue of assignments, so that the next time Teddy contacts the organization, he receives these assignments. It's almost like an inbox that only reveals its queue of messages when someone makes inbound contact. You already see this pattern used by cellphone companies: when you call, they remind you that you're due for a device upgrade.
  With inbound calls this pattern can be used to deftly avoid prolonged calls. For instance, if someone who recently placed an order calls in, there's an assignment at the top of their queue to let them know: "I see you ordered something from us earlier this week. Good news—your order has shipped! Would you like the tracking information?" A personalized experience like this saves the user time and engenders confidence in the IDW. PROMISES AND ASSIGNMENTS is an amazing pattern because it allows you to go further without annoying people. They've already reached out to you; meet them with some useful information that can save them time.

- **COORDINATE.** This pattern is meant to get several participants working together around a particular goal. This might be used to schedule a meeting or gather shared input. This is a more complex pattern that will often have sub-patterns like CHASE, TRACK, and TRANSACT working together.

- **SHARE.** This pattern is designed to share information with people who need it. This proactive pattern helps disseminate information in a relevant or contextual way. Flows that use SHARE will send out messages or links bearing useful information.

- **TEACH.** This pattern teaches users how to do something, often launching from Q&A. The purpose is to provide a series of lessons and/or instruction, which can occur in a single session or across multiple sessions.

- **PREDICT.** With this pattern the machine uses past interactions and contextual data to predict what a user might be trying to do, often suggesting possible outcomes, such as by saying: "This option is statistically more likely to produce your desired outcome." By reviewing all available data to predict possible outcomes, PREDICT eliminates unnecessary steps to make the conversation as efficient as possible.

- **PROCESS MINING.** A machine can be trained to analyze data with the objective of identifying patterns, inefficiencies, and opportunities in both current and historical processes and events.

- **ANOMALY DETECTION.** This is a process mining pattern for detecting anomalies in data, such as events or deviations from what is standard or expected.

- **ANOMALOUS ABSENCE DETECTION (AAD).** An especially valuable type of anomaly detection involves regular evaluation of data with the particular goal of detecting something humans aren't typically good at spotting: the absence of data. Essentially, the machine identifies patterns and meaning in the absence of data, and treats each absence as an event—which in some cases could amount to identifying missed opportunities or opportunity costs.

- **MACHINE-IN-THE-MIDDLE (MITM).** This is a relatively simple pattern that involves a live agent handing off a user to an IDW in order to perform a task that a machine can do more efficiently. For instance, if you've been talking to an agent to arrange specific details of a purchase, that agent can then hand you over to an IDW to collect payment information (maybe a text pops up on your phone requesting your credit card number or a photo of the back of your card). Another example: you email a service-provider with a query and an IDW tasked with reviewing emails sent to their support team detects missing information. You get a follow-up email from the IDW requesting your account number and the service address, so that when the agent pulls up the ticket, all of the information is there. This unburdens the agent from having to do additional follow-ups and helps your request reach a faster resolution.

  This is also an important pattern to keep in mind because it demonstrates that there are plenty of creative ways to create automations that don't require APIs or extensive integration.
  MACHINE-IN-THE-MIDDLE can simply be that an IDW that receives an inbound email, uses NLU to see review the content, and determines whether there's missing information. The IDW can reply right away asking users for missing data—no integration necessary.

- **HUMAN IN THE LOOP (HITL).** Automation is only as good as the data at its disposal. In this powerful pattern, humans provide data to help train the IDW—and there are endless conversations or variations of conversations that humans can help with. Flows that incorporate HITL reach out to their humans-in-the-loop on different channels—whether it's a call center, chat channels, text messages, or collaborative tools like Slack—to get the needed information; in turn, the flows update their knowledge bases and skills. In other scenarios—as has been previously covered—when the IDWs get stuck on a problem, they are guided by humans with what to do next. Team members can either feed the IDW information for machine learning or script their interactions in a particular way.

- **HUMAN-CONTROLLED OUTCOMES (HCO).** Hyperautomation needs to be human led at every level. Machines' abilities to make efficient decisions on their own will continue to improve, but it's crucial to keep people in control of outcomes. People don't want to live under strict orders from machines — even if those machines are designed to maximize efficiency. But people will likely look forward to interacting with machines that regularly offer efficiency-improving suggestions.

- **METACOGNITIVE BEHAVIOR.** This is more of an overarching pattern—one that embraces and reinforces many of those described above. The idea is to create a pattern of awareness within your ecosystem so that, while IDWs are learning individual skills, they are also managing their overall learning. It's one thing to learn a skill—dogs, for instance, do it all the time—but it's another thing to be actively aware that you are learning (and to subsequently project-manage that learning). For our purposes, this could be as basic as having an IDW check in about learning new skills set along a timeline. It could also include higher-level functionalities, like having an IDW check in with suggestions based on user queries. (E.g., "I've noticed many users are calling to request password resets. Is this something I can learn to do?") You could also seek out new tools for your ecosystem, and then vet them based on reputation or user ratings.