Unpacking an Expert's Tacit Knowledge

Clark Aldrich proposes fifteen questions that will help instructor-experts establish the *specific details* of their tacit knowledge, the details that set them apart from beginners.

1. Can you think of an experience that epitomized the subject matter? (This could be a real-time meeting or an event that unfolded over weeks, months, even years.)
2. In that situation, what options did you consider to be available to you? Can you go step by step through the actions you took? Would a "naïve" or inexperienced person have acted differently?
3. What things might the naïve approach fail to account for?
4. What are some clues that informed your knowledge of the situation? What did you see immediately, and what did you have to look for? How did you go about looking?
5. What would you have considered to be a successful outcome? Was that the actual outcome?
6. What were you looking for to check whether things were going well? What were you looking for to check whether things were going wrong?
7. What were the "maintenance" or routine activities you had to perform (even down to body language)? What would have happened if you hadn't performed them?
8. When did you know you'd been successful (or hadn't)?
9. If the experience involved other people, what was each person's best-case and worst-case outcome? What were their strategies? What did they do?
10. What are three to five other ways someone could approach the situation and still have a reasonable chance of being successful?
11. What are three to five of the highest-level metrics you were monitoring? Time? Commitment? Alignment?
12. What tradeoffs were you willing to make?
13. Can you graph the high-level metrics over the course of the experience?
14. What were the inflection points for each metric?
15. How did your actions impact the high-level metrics? What else impacted them? (Be as specific as possible.)

WEB COURSEWORKS, LTD.

Creating Serious Games and Simulations: A Quick & Dirty Guide

Clark Aldrich

Web Courseworks, a national eLearning development firm, and Clark Aldrich, a renowned simulation designer and author, have partnered to design and produce games and simulations. This white paper outlines some of the processes we use to craft learning games and simulations.

March 2010

In this white paper:

- Learn how to craft a simulation or learning game from a combination of explicit and tacit knowledge.
- Chunk out the design and development process and estimate time and complexity.
- Learn how to leverage good game design to improve your simulation or learning game.

What follows is methodology and identified best practices to produce a sim. Let us first look at a framework to get you through the design process in four predictable steps.

Step 1. Collect Explicit Knowledge

The first step in designing a sim is to collect all of the explicit knowledge, much of which will already be documented: top-down patterns such as established analysis, best practices, and rules.

In this step traditional educational content and linear material, such as courses and curricula, books, reports, famous or inspirational quotes, rules, and policies are very helpful. They also serve to set a scale for what the sim will and will not cover.

Step 2. Identify Tacit Knowledge

The second step, after all of the traditional rules and analysis are collected, is a daunting one: as best you can, uncover the experts' tacit knowledge.

What is tacit knowledge? It is all of the tiny relationships that an expert understands, but often has trouble putting into words. It is experience-based knowledge triggered by certain key situations. These are situations that the expert has experienced so many times that he or she knows what actions to take to achieve the desired results, even if he or she could not put words to when, where, or why the actions work.

Where explicit knowledge is top-down, gained through formal study of curricula and rules, tacit knowledge is bottom-up, gained through experience and experimentation. How does a manager know when to pull back from pressuring an employee, a writer know when to separate a long sentence into two

Collecting Explicit Knowledge: An Example of Step 1 Output from Composting

Explicit knowledge is top-down patterns such as established analysis, best practices, and rules. If you were building a simulation about composting, you would collect all of the common advice established by experts such as:

- Don't throw in dairy or meat
- Turn your pile every few weeks
- Mix in grass clippings to keep the nitrogen at the right level so it doesn't smell
- And people compost to reduce their impact on landfills and improve their land

sentences, or a cook know when to sauté the onions for 30 more seconds? This is tacit knowledge.

Tacit knowledge falls into the simulation framework of actions, systems, and results. (See The Complete Guide to Simulations and Serious Games for an in-depth discussion.)

Tacit Knowledge A: Actions

An expert's tacit knowledge includes actions. Examples of actions are: yell, beg, put peg A in groove B, invest money, run. To uncover these actions, here are the biggest questions to ask:

- What actions will an expert consider taking?
- What non-optimal actions might a non-expert take?
- Can the actions be defined very specifically, down to exact quotes or levels of magnitude?

The sim should present the learner with options: both the target actions that an expert would do, and incorrect actions that a non-expert might take. Learning happens when the learner starts to correlate the actions to systems and results.

For example, imagine we were creating a sim around end user computer security. Some of the actions available when a user gets a forwarded email are:

- Follow an embedded link to a web site
- Open an attachment
- Forward the email to a friend
- Install a suggested program
- Try to figure out if the email is legitimate
- Delete the email
- Report the email to their manager or IT department

As you might guess, the last two actions are the target actions that an expert would take.

Importantly, a sim designer may have to uncover the context in which the target actions are done. For our end-user computer security example, most people would recognize the target action when the options are presented out of context. However, people face these situations in very different contexts: while they are focusing on doing their job, managing their personal

Identifying Tacit Knowledge: An Example of Step 2 Output from Composting

Actions: put different kinds of food in compost, turn compost, shovel out and spread compost, put in other organic matter, cover pile, buy barrels, use mixing tools, sift compost, throw out food as garbage, design compost area.

Systems: rain washes through compost, food breaks down with aeration in about a month, food breaks down without aeration in about a year, garbage costs money per pound to put in a landfill, exposed vegetables will mildly attract critters, earthworms can aerate dirt.

Results: rich soil, rotting smell, less garbage sent to landfills, yellow jackets, great vegetables, critters.

life, or entertaining themselves and relaxing. These "life" actions might have to be worked into any final set of actions available in the sim.

Tacit Knowledge B: Results

The second category of tacit knowledge on which we need to focus is results. For results we ask what success and failure are. Is success all or nothing, such as the accomplishment of a mission? Or are there three or four things that a person is trying to balance and grow? Or, is success in the sim (as well as from the sim) the ability to consistently apply an increasingly complex set of competencies?

Again, we look at both targets and contexts. The target result for computer security might be a smooth IT environment, as opposed to a massive virus infection. But the contextual results are just as important. For example, people need a smooth IT environment within the larger context of their jobs, and for that reason never opening any attachments is not a viable option, even though it would be the safest way of to ensuring a smooth IT environment.

For other common simulations, being an ethical person or a great leader also happens only in context. We want the target results to occur in context.

Identifying the results of failure is more interesting, more important, and more counter-intuitive for most instructional designers than identifying success.

- What are the various types of failure one can experience
- What are the situations that lead to these failures?
- What are the immediate wrong things to do?
- What are long-term failures?

Any given user should not see most failure states that have been created. The one they do see will be targeted to their individual weakness and should align with real life.

Tacit Knowledge C: Systems

The final set of relationships in tacit knowledge is systems. Systems are what get between actions and desired results. If the

The Order of Development Matters (A Lot): the Problem with Starting with Step 2's Tacit Knowledge

While each of the three steps should inform the other steps, the order is important. If you put Step Three before Step Two, you will get dramatically opposite effects.

Starting with Step Two often occurs when a researcher or subject matter expert starts an effort. Identifying the little relationships (situated knowledge) without the framing of the best practices (explicit knowledge) is a staggeringly complex activity. It satiates the purists, but it takes huge amounts of time and overwhelms all but the most intrepid. Stakeholders who are not intimately involved with the project are mystified by this process.

Projects that start here seldom see the light of day. Even if they do survive, there is much wasted effort.

collection of all sets of tiny relationships is an iceberg, the systems are the part of the iceberg that is underwater – often a huge hidden mass.

Here are two quick examples:

- In chess, a player wants to use the pieces' permitted actions to achieve the result of checkmating the other player's king. However, the systems of rules and positions and the activities of an opponent on the board need to be navigated and overcome.
- In leadership, we may want to build a great team, but the rules of accomplishment, personal egos, motivation, and reward need to be navigated.

Some questions might be: are there processes or mazes that have to be followed? Are there opponents that are striving to keep the person from being successful? Are there cycles, balancing loops, or feedback loops? Are there delays? Are there mathematical relationships? Are there hidden processes that others are following?

Finding "Tiny Relationships"

Many of these relationships are so simple that it feels absurd to even capture them in a document, but their power comes from their rigor, volume, and integration. You have to be a detective here: grill subject matter experts, listen to podcasts, watch videos, and generally obtain as much information as possible.

Step 3. Find the Closest Existing Simulation or Game

We have already identified the high level rules (explicit knowledge) and the mounds of tiny relationships (tacit knowledge). Now, find an existing toolset, game or sim genre, or microcosm that comes close to the framework or spirit of some or all of what you want to accomplish. If at all possible do this step *after S*teps One and Two (see sidebar), instead of making the common mistake of doing it first.

The Order of Development Matters (A Lot): the Problem with Starting with Step 3's Toolkit and Genre

Starting with the identification of the genre or engine (Step Three) and filling in the blanks is a common worst practice. I often see this when a vendor has a pre-built engine they are using for a new project. The results are quick (weeks instead of months), cost effective, and efficient. The course is spit out on time.

The only problem is that the content is flat. Designers end up reskinning rather than teaching anything of note. Two or three different programs, ostensibly covering different topics, starting from the same engine, all look the same. More importantly, they basically teach the same thing. We are seeing this in abundance with sims in Second Life.

From a business perspective, this makes sense for them – the vendor's internal cost, time frame, unpredictability, and quality of talent needed are five to ten times greater if they are creating a new engine instead of using an old one. But it can result in a substandard or forced student experience.

Borrow an established format if possible

If an engine exists, such as Second Life or Adventuremaker, figure out how to use it. This can save more than 80% of the development time. Regardless, use the gameplay and level design conventions. In many cases, you will also draw models from other genres as well, glomming them together. Even the narrowest toolset allows for importing great ideas.

There are times when there are no appropriate games or sims, let alone toolsets. In these cases, find one perfect example or microcosm that can serve as the model for the interaction.

Step 4. Synchronize

Now, at stage four, we have to bring everything together. Use the top-down rules in Step One to organize the tiny relationships in Step Two, and then use the genre from step Three to frame everything. Work from these three corners to the middle. Ultimately, all three should converge, even if there is fear at first that they won't.

Reconciling Broad Rules and Tiny Relationships

During this process, we start seeing plenty of places where the broad rules from Step One and the tiny relationships from Step Two do not necessary align.

One example is when you are given a list of different possible successful approaches with superficial illustrative examples. For example, the old training might read:

> *To influence someone, a leader can tell a person what to do, but the leader can also bribe, threaten, appeal to their sense of purpose, ask them as a favor, or make a logical case for a request. To illustrate: consider a documented case where a CFO was asked to postpone her retirement, and the new CEO was successful because he appealed to her loyalty to the company.*

This may be a sufficient for a PowerPoint slide, but like philosophy, it begs more questions than it answers for a sim designer. Things that need to be reconciled include:

Synchronization of Content: An Example of Step 4 from Composting

My goal is a thriving ecosystem, so I might borrow from an existing model that creates a thriving ecosystem: SimCity, for example, or Roller Coaster Tycoon. I might use quality of life, cost, and environmental impact as some core metrics the player tries to optimize. I might create a house area, a compost area, a garden area, and a garbage area, and have people be able to move stuff between the four. Finally, moving away from these genres, I might zoom in and allow people to create and modify their own composting structure.

- Why did the expert (the CEO) use that approach? Was that his favorite influence strategy? Had that worked before with the CFO? Was there some contextual or systematic tell that clued him into this approach?
- Did the CEO consider two or three different approaches, and what were the criteria that won out?
- Did the CEO switch approaches midstream, and if so, why?

At a higher level, is there a common underlying model of tacit knowledge that aligns most of the identified approaches? When sifting through the body of linear content, you will find many situations where experts offer different and contradictory advice.

A classic contradictory construct: Are you turning the other cheek (good) or are you appeasing (bad)? As with the above situation, a goal is to find common mechanics that allow for both.

Other Steps in the Synchronization Process

As one closes in on a final design, some tough questions have to be answered.

How broadly can the identified actions be abstracted? Recall for a moment our list of actions from the computer security example:

- Follow an embedded link to a web site
- Open an attachment
- Forward the email to a friend
- Install a suggested program
- Try to figure out if the email is legitimate
- Delete the email
- Report the email to their manager or IT department

All of these actions can be generally abstracted into three core actions: acting on the incoming request, probing the request, or rejecting the request.

This is critical, as most sims work best in real time, where the computer does not wait for the student. Ideally a few actions are applied repeatedly, in different orders and with sensitivity to timing. Abstracting actions can increase the applicability of the sim to wider groups.

Another type of problem we have to answer: how does the sim respond to little failures? Being inappropriately aggressive to a subordinate, for example, is a bad idea in a leadership sim. But does it stop the whole sim? In real life, plenty of successful people have little slips. Are little failures cumulative? Jumping ahead to Step Three for a moment, arcade games often had a “three lives” model. Is that appropriate?

At the end of the process, you may have a few outlier rules from Step One that fall outside of the system and level designs that you have created. These rules may still need to be included to

satisfy your sponsors. Hopefully this is minimal, but you might convey this content through traditional pedagogical technique such as slides or pop-ups. You know you have done a good job when all three perspectives support each other rather than grind against each other. Often you will gain unique and industry-valued perspectives through this process.

The Three Trimesters of a Serious Game Development Process

Of course, the design process has to fit into a larger serious game development process. I will now zoom out a bit and look at the rest of the process.

How long does it take to create a Serious Game?

Before we dig in, let's talk a bit about end-to-end time frames. There are two sides. On this side, the most effective sims of the next five years will be single-player, will be Adobe Flash-based, will take about one to two hours of student time, and will take about nine months to create from scratch

On the other side, most corporations want something delivered within about three months of signing a contract. These two realities are not incompatible, thankfully. If you have to, you can get something out the door in weeks by manipulating some significant, cumulative modifiers. Here are key modifiers that can decrease and increase the nine-month development time:

Creating Strategic and First Person Perspective

The most effective sims use two or more parallel and mutually-reinforcing perspectives. This approach is consistent with generations of computer games and flight simulators. These have traditionally featured a first person perspective and a strategic perspective. The pro-typical example is of a driving game, where the screen is used to show the world from the driver's perspective looking out at the highway and other nearby cars, while also showing a top-down perspective on the entire track with all competitors. Players made decisions based on both perspectives simultaneously.

The first-person perspective presents the actual decisions that the student will see and make in the real world. This often involves interpersonal conversations. The strategic perspective presents the "big picture" and involves a visualization of a system and interactions often invisible in the real world.

Dev-time Decreasers		Dev-time Increasers	
- 25%	Single player instead of multiplayer	+ 60%	Multiplayer in addition to single player
- 70%	Lightweight, browser-based mechanics	+ 100%	3D, client-installed "Game-Like"
	Reuse Established Approach:		*Create Totally New Genre:*
- 60%	Complete adherence to an existing genre	+ 30%	Completely new genre
-50%	Engine already exists	+ 20%	Flexible and reusable architecture

The time for a serious game usually falls into three trimesters: create, code, and calibrate. The above time modifiers tend to impact all three trimesters evenly. Now, let's look at the three trimesters in some detail, including key roles and responsibilities, and keeping mind that the three trimesters tend to overlap.

Sim Development Trimester One: Create

In the first trimester, the sim is designed using some of the techniques discussed above. The goal is to produce a great design document, between 30 and 50 pages long. The lead designer dives deep into the content, often becoming an expert.

Also in this trimester the learning objectives and requirements are formalized, often using people in the role of a client liaison and program sponsor. The look and feel is nailed down, hopefully with the work of a good graphic designer. Any technical decisions, including media, authoring environments, engines, and end-user requirements, are established. Steps also have to be taken to set up Trimester Two.

Sim Development Trimester Two: Code

In the second trimester, two or three programmers/coders program the material in the design document. They produce much of the core sim engine itself. They provide links to the fluid content, such as graphic files, videos, sound files, text, and entire level designs and sim flow, using industry standard media and XMLs. The program sponsor, lead designer, graphic designer, and client liaison are peripherally involved; making decisions and helping flesh out the numerous parts of the sim engine that need refining. Near the end of this process, the lead

designer begins inputting as much of the final content as possible. About 70% of the project budget is spent in this stage.

Sim Development Trimester Three: Calibrate

In the final trimester, the lead designer finishes inputting content into the engine, and the entire package is put in front of target audiences by the program sponsor. The programmers/coders need to be available to make core engine changes, but even more so the lead designer and client liaison have to refine the fluid content. Finally, there can be integration work with the LMS or database.

Final Thoughts

Things have never seemed harder for those tasked with developing the skill sets of organizations. They have to deliver content and sometimes entire curricula with coaching and certification. They have to do so with minimal costs in development and delivery dollars, student time, and student disruption.

The good news is that simulations and serious games can instruct more in less time and at less cost. The most successful organizations will either have an internal sim development capability or partner with an external vendor that does. I hope following these steps and processes makes the implementation a bit easier and more predictable.

About Clark Aldrich:

Clark Aldrich designs and builds educational simulations for a wide range of corporate, academic, government, and military clients. He is also the author of four books, including his most recent The Complete Guide to Simulations and Serious Games. His blog is at http://clarkaldrich.blogspot.com, and he can be reached at clark.aldrich@gmail.com.

About Web Courseworks:

With offices in Washington, D.C. and Madison, WI, Web Courseworks designs and develops custom Flash-based learning games and simulations for corporate training, K-16 education, and non-profit outreach. Web Courseworks has also entered the association LMS market with a hosted learning management solution called Coursestage™. Visit www.webcourseworks.com or www.coursestage.com for more information.