FTV226626

Shotgun for Production Management in LAIKA’s Animated Features

Tony Aiello
LAIKA

Learning Objectives

- Understand how Shotgun has been used to track and refine the production of LAIKA’s films
- Understand the workflows involved in stop-motion feature film production
- See how LAIKA has automated aspects of the design-and-make process for stop-motion film production
- Appreciate and understand the parallels between feature film and industrial/factory production management tracking

Description

LAIKA is a stop-motion feature animation studio located near Portland, Oregon. Autodesk’s Shotgun project management software has been used to organize and automate aspects of the production of all its feature films over the past 10 years, from Coraline to the forthcoming Missing Link. In this talk, I will present how LAIKA tackles the design and manufacturing of the assets that show up on the big screen, how we use Shotgun software to track our production data (from scheduling to final delivery and beyond), and how we have built an efficient production pipeline using Shotgun as a platform. LAIKA exists at an interesting junction between visual effects and manufacturing, producing physical assets for a digital final product. As we go through our workflows, we will also reflect on the commonalities we have seen in how to create both physical and digital assets efficiently.

Speaker

Tony Aiello’s career has centered around hardware and software systems development and integration, with a particular focus on human factors and usability. Having previously applied these skills in defense R&D then machine vision and industrial automation, for the last decade he’s been living the dream in feature films, with production technology credits on Coraline, The Boxtrolls, and Kubo and the Two Strings. As a Senior Engineer in LAIKA’s Production Technology department, he’s been building production management systems, many of which have involved pushing Autodesk’s Shotgun to new extremes.

www.linkedin.com/in/tonyaiello
Introduction

Fueled by the vision of its President & CEO Travis Knight, the animation studio LAIKA was founded in 2005. Located just outside of Portland, Oregon, LAIKA was awarded a Scientific and Technology Oscar® plaque in 2016 for its innovation in 3D printing. All of LAIKA’s four films: Coraline (2009), ParaNorman (2012), The Boxtrolls (2014) and Kubo and the Two Strings (2016) were nominated for an Academy Award® for Outstanding Animated Feature. Kubo and the Two Strings also won the BAFTA for Best Animated Film and received an Oscar® nomination for Visual Effects. LAIKA’s next film, Missing Link, will be released April 12, 2019 by Annapurna Pictures. The film boasts a roster of international stars including Hugh Jackman, Zoe Saldana and Zach Galifianakis.

LAIKA combines the best traditional stop motion artistry with cutting edge technology like 3D printing, robotics and computer graphics to tell original stories in a way people have never seen before. This results in hybrid processes and a unique style. LAIKA tries to capture as much imagery in-camera as possible, which means that we’re physically constructing much of what you finally see on screen, but nor do we shy away from making our movies as compelling as possible through digital artistry. Tracking both the digital as well as physical production of thousands of items from as small as a miniature button to as large as a 50-foot-square samurai castle set, and coordinating the efforts of all those people responsible, over the course of 3 to 4 years, is an enormously complex logistical operation for which there is no blueprint.

To meet that challenge, LAIKA deployed and began investing in Shotgun in early January 2008:

![Figure 1. EventLogEntry #2](https://shotgun.laika.com/detail/EventLogEntry/2)

![Figure 2. EventLogEntry #100,000,000](https://shotgun.laika.com/detail/EventLogEntry/100000000)

Now more than a decade later and almost exactly one month ago, LAIKA’s current production Shotgun site hit its 100 millionth EventLogEntry:

Shotgun provides a customizable user interface on top of a powerful and scalable data storage back-end, with capabilities for integration into other systems. This allows Shotgun to appear as a collaborative media wrangling tool to artists, while also a task and resource scheduling system to coordinators and managers, and yet again as a tracking dashboard for department heads and executives, all in one. Over the years and the movies, our use of Shotgun’s features has continued to expand into more and more of LAIKA’s workflows.

So, “how do we make a movie?” In the beginning, there’s just a concept: of some characters, in some locations, and something happens. Developing and producing a stop-motion feature film is like the oft-repeated joke: “how do you eat an elephant? One bite at a time.” Let’s start by looking at the elephant-eaters, who are all represented in our Shotgun install.
The Players

There are top-level Departments, such as Art, Puppets, Rapid, VFX, Animation, and more. Within these Departments are the sub-departments, what we call Shops. These include the Set Shop, Model Shop, Scenic and Paint within Art; Armatures and Molds and Hands in Puppets, and many more. We use Shotgun’s Home Department entity for the top-level departments, and its Group entity to represent the shops within:

**Figure 3. LAIKA DEPARTMENTS IN SHOTGUN**

**Figure 4. SHOPS WITHIN DEPARTMENTS**
Departments and Shops Workflow
Like any large-scale construction effort, our production process is simply that each Department takes a chunk and iteratively breaks it down into smaller pieces, with increasingly specialized teams taking the pieces further, until the final product is assembled. That flow looks like this:

**Figure 5. Departments and Shops Workflow, Color-Coordinated with Shotgun**

Generally speaking, every arrow in the diagram represents Shotgun “workflow” entities like Tasks and Notes, while the ovals are “products”, represented by other Shotgun entities such as Sequences, Shots, Assets, and Versions. Every Department follows the same basic flow:

**Concept ➔ Look Dev ➔ Design ➔ Prototype ➔ Previz ➔ Build ➔ Deliver ➔ Maintain**

You could think of the data in Shotgun as a giant web of connected things. The connections in the web are “fields” that point to other types of things or data. Sometimes the connection only goes one way ➔ and can only link to a single thing (a Shot can only be in one Sequence for example), while other times, the connection is two way ↔ and can link to multiple things (a Version can be in as many Playlists as you want). We make pages in Shotgun to see pieces of
that web. The pieces of the web are the things each workflow produces and the schedule of resources required to produce those things. Each Department, and often each Shop within a Department, has pages that model their workflows and represents how they connect to others.

**Asset Hierarchy - Sets**

```
Sets
  ↓
Set Builds
  ↓
Set Parts
    ↓
Tasks
```

**Asset Hierarchy - Characters**

```
Character
  ↓
Puppet Builds
  ↓
Tasks
```

We need to track the resources and schedule the activities to produce them, which are then required by other scheduled activities – ultimately to produce the approximately 150,000 images that make up the movie. Those activities are represented by Shotgun Tasks. Let’s start talking about Tasks with the most basic ways we group and create them.
Shotgun Pipeline Steps
A Pipeline Step in Shotgun is a grouping mechanism – it’s a way to collect and relate a bunch of Tasks and ask questions about them as a group, like what’s the earliest one, what’s the latest, what percentage are complete, etc. In most primarily-VFX studios, a Shotgun Pipeline Step is often used to correspond to what we call a Shop – so they might have a Modeling Step, a Layout Step, a Rigging Step, a Surfacing Step, etc. With the number of Departments and Shops we have at LAIKA, we felt per-Shop-steps was a bit too granular. So, we correspond a Department to a Pipeline Step, then delineate Shop-level Tasks using the Task ‘Type’ field or a custom field “Assigned To Shop”.

Shotgun Task Templates
Shotgun supports creating Task Templates, which create a set of default Tasks when a Task Template is applied to a product-type entity. We make extensive use of Task Templates for creating Tasks that get automatically linked to Shots and build Assets.

**FIGURE 6. PIPELINE STEPS**

**FIGURE 7. TASK TEMPLATES**
The Process

Every movie starts with a concept. That concept gets refined over what can be a long time – months or even years – during which preliminary designs are developed and a script gets written. While LAIKA tracks a certain amount of that early development process in Shotgun, herein we’ll skip ahead to green-light: the point at which the production departments are told to get started and make the movie. The process after green-light can be summarized as:

Scoping → Bidding ↔ Scheduling ↔ Producing ↔ Tracking

Scoping

Some initial assumptions have to be made and some analysis performed. Typically, this starts with the script. Scripts at LAIKA have been both sourced externally and developed internally.

Basic Assumptions

- 90 minutes of screen-time (movie length)
- Approximate pace of 4 seconds of screen-time by one animator per week
  - so let’s do some quick math:
    - 90 minutes = 5,400 seconds / 4-per-week = 1,350 person-weeks!
- Approximately 2 dozen animators
- Approximately 2 to 3 shooting stages per animator
- 90 weeks of animation and shooting, also known as Production
- “some” number of weeks of Pre-Production – building physical and digital components that must be ready for animation and shooting to begin
- “some” number of weeks of Post-Production – finishing CG work and marketing

Script Analysis

We require scripts in Final Draft format and with a certain set of naming conventions for scenes and characters and locations. The conventions and open XML format enable integration with Shotgun right away, with a tool we’ve written called script_splitter. script_splitter is written in Python and relies on Shotgun’s API.

```
usage: script_splitter [options] <scriptFile>

options:
- -h, --help     show this help message and exit
- -b, --breakdownMode  Create any characters in the script that are not
                      already in Shotgun
- -d, --dryRun     Dry run, don't upload anything to shotgun
- -l, --highlightedinonly  only load dialog that is highlighted
- -o charOverrideFile, --charOverrideFile=charOverrideFile  Character name override file e.g. map 'Child's Voice' to 'Kubo / MB'
- -s, --shotgun    Upload results to shotgun
- -t, --nosceneDialog Include character names and dialog in scene
descriptions uploaded to shotgun
- -v, --verbose    print verbose messages
- -q, --quiet      suppress messages
```

**Figure 8.** SCRIPT_SPLITTER IS COMMAND-LINE ONLY, NO GUI
Functionally, \texttt{script_splitter} is very important, because what it does is extract all the Sequences, Scenes, Characters, Locations and Sets, and even DialogLines, from the Final Draft script and create all those entities in Shotgun, relying on those established naming and typing and link conventions. Before running \texttt{script_splitter}, the Production Technology department simply creates a Project entity in Shotgun, to which everything else will get attached.

The characters and sets at this point are Shotgun Assets, and are considered the “conceptual” entities – database entries to which we’ll publish concept, design, and reference artwork.

**Broad Strokes Schedule Estimate: “The Oneline”**

With these initial conceptual entities now in Shotgun, we can start planning the work. In this estimation phase, we use the script's Scenes as a measurement entity. We create essentially a single start-and-stop schedule for each Scene, which we represent in Shotgun with one Task per Scene. We call these Tasks, and set their Shotgun Type to, “Oneline” Tasks, simply because each Scene gets represented on one line. These Oneline Tasks are created with another command-line tool, named cleverly enough \texttt{create_oneline_tasks}. The essential features of Oneline Tasks are:

1. they are schedulable – they have start dates and durations and end dates;
2. required resources can be attached to them – “this scene requires Kubo and Monkey in a cave, that scene requires the evil sisters on a boat”, et. cetera.
3. they support the other resources required, such as “generic” scheduling animators (“bot” HumanUsers) and the shooting stages, called Units, which we represent with a Shotgun CustomNonProjectEntity.

The duration of a Oneline Task is calculated as a proportion of the length of the scene in the script using the animation pace assumption. In other words, we calculate a simple ratio: "this scene covering 1 page in the script will take 12 weeks to animate and shoot." \texttt{script_splitter} uploads the page length of a scene to its Scene entity in Shotgun, then \texttt{create_oneline_tasks} calculates each Oneline Task Duration from its correspondingly-linked Scene’s page length. Oneline Tasks are Linked to their Scenes with Shotgun’s default entity field.

Production Management’s job then is to distribute these Oneline Tasks over the estimated 90-week shoot onto our available shooting stages, in such a way that the animators are kept fully busy and yet the number of instances where we would have to duplicate a physical character or set or prop or RP face-kit are minimized.

The outcome of this Oneline creation process gives the other departments the deadlines they need to hit – the dates on which the first build of each physical asset must be ready. In this way, we’re creating an estimated shooting schedule so as to employ Just-In-Time planning for the assets to be constructed.
The Oneline also shows how many actual Builds of any given Character or Set will be required. Unlike a CG asset, a physical asset can’t be in two places at once. So, viewing the Oneline in a Gantt view and doing overload-analysis on the resources connected to those Tasks tells us the number of physical first-builds and duplicates we’ll need to make, which become the required resources for finer-grained scheduling later. In *Kubo and the Two Strings*, because Kubo is the hero and thus in almost every scene of the movie, we had to build nearly 30 Kubo puppets!
We represent the physical products with Shotgun Assets, with yet more Asset Types: “Puppet Builds”, “Set Builds”, “Rapid Cores” and “Rapid Faces / Face Kits”. In fact, everything we build, practical or digital, is represented in Shotgun by an Asset entity and distinguished by its Type field. Thus, we make extensive use of Shotgun’s Detail-Page-per-Asset-Type feature.

![Detail Pages for the 'Set' and 'Character' Asset Types](image)

**Figure 10. Detail Pages for the 'Set' and 'Character' Asset Types**

Now the builders need to estimate their workloads to meet those deliveries and dates.

**Bidding**

Every fabrication Department at LAIKA needs to tell the producer of the movie what it’s going to take and cost for them to do their job. We’ve combined many tools with Shotgun to do that.

**Excel Complexity Bid Template**

Production Technology supplies each Department with an Excel spreadsheet exported from Shotgun with all the Builds required (as indicated by the Oneline) listed down vertically. Across the top, there are columns for each Shop / sub-process involved, as well as a Complexity rating for each Shop. Complexity is just a number from 1 to 5. This feeds into a sub-sheet within the Excel file for each Shop, mapping the complexity bid number to a set of bid durations for each process within the Shop – for example, a complexity of 1 might be “2 days of molding plus 3 days of casting” etc., while a 5 may be weeks of such time. The Excel template rolls all these duration values up into the Shop Tasks based on the Department simply entering complexity bid values. This is
obviously a lot faster than typing similar numbers of days over and over again. The Department completes the Excel file by entering default durations for complexity values for Shops and their complexity bids for each of the Assets they’re asked to build.

Shotgun TaskTemplates and Microsoft Project Templates
The actual work-breakdown-structure (WBS) defining the dependency relationships of the Tasks involved in constructing a build gets modeled in Shotgun as TaskTemplates and also in Microsoft Project. The Tasks are exactly matched between the Shotgun TaskTemplate and the MS-Project template. The promise of Shotgun TaskTemplates is that when it comes time to create the actual Tasks for the Builds in Shotgun, the Department would be able to just apply the appropriate TaskTemplate to each Build.

Unfortunately, it’s not quite that easy. When you consider that the Production Departments are bidding Complexities for each of the different Shops involved, the number of possible TaskTemplate permutations explodes exponentially. So, we needed a way to dynamically populate the actual Task durations at TaskTemplate-application time based on the complexity bids. Further, Shotgun currently has another scheduling limitation: it currently lacks Task-to-Task WBS relationships other than Finish-to-Start, including percentage offsets, “this task starts 50% of the way through its predecessor.”

This is where MS-Project comes in. Production Technology employs another CLI Python app we’ve written excel_to_project to extract the per-Build-per-Task duration values from the completed Excel spreadsheet and match them up with corresponding Tasks in the MS-Project template. There can be two or more MS-Project templates – for example, for a “first build” which typically has lots of testing tasks and approval milestones, or for a “duplicate build” which is after the first-build has been fully approved and no longer needs approval or testing milestones. excel_to_project replaces the durations for the tasks in the MS-Project template with the actual bid durations from the Excel spreadsheet, and the need-by-dates from querying the Shotgun Oneline Tasks, finally generating a text file of build Tasks with MS-Project-calculated WBS relationships for every Build asset in the bid. That file then gets imported back into MS-Project again, which does all the complex date-math, yielding a construction schedule for all Builds complete with bid durations and start and end dates!
Upload to Shotgun

But we’re not quite there yet – those Tasks are all still in MS-Project. The final step is to get them into Shotgun, using yet another of our apps, project_to_shotgun. The full-schedule MS-Project file is exported back into text format which project_to_shotgun reads to create Shotgun Task entities with all the dates and durations that MS-Project calculated for us with its full range of WBS features. Then it applies the appropriate Shotgun TaskTemplate for each Build, and finally updates the newly-created Shotgun Tasks with the MS-Project calculated dates and durations.
**Figure 13.** *PROJECT_TO_SHOTGUN UPloading the SET BUILDS SCHEDULE*

**Figure 14.** *SET CONSTRUCTION GANTT CHART AFTER UPLOAD, EXPORTED FROM SHOTGUN*
After a few minutes, thousands of Tasks have been created in Shotgun. Each Department then takes over maintenance of their schedule exclusively in Shotgun to perform schedule adjustments. And fortunately, Shotgun does make it easy to move a block of selected Tasks all at once, maintaining their relative date relationships even without linked WBS dependencies. Adjustments are necessary because at this point the schedule was generated only based on estimated Oneline Task dates. The real dates required by Stage Production – that is, the Animation and Camera Departments – are still to come, from the process of Shot scheduling.

Multiple Scenarios
And of course, there’s always the question of “what might things look like if we changed the schedule / number of builds / number of people / number of hours in a day / etc.” In other words, multiple possible scenarios. Obviously with MS-Project, this is as easy as working in another copy of the file. But we’ve become victims of our own Shotgun success – no one here wants to use MS-Project anymore (except Production Technology… actually, we don’t either). So, we needed to invent a way to have multiple scenarios of thousands of Tasks all within Shotgun.

Multiple Task scenarios isn’t a natively-supported feature in Shotgun at this time, but it turned out to be not too difficult. For each Department’s Pipeline Step, we added an additional Pipeline Step named the same but with “Planning” appended – see Figure 6. Pipeline Steps. Then, we wrote a python script `sg_planning_tasks` connected to a Shotgun Action Menu Item.

![Figure 15. Planning Tasks and Other AMI's](image-url)
The Planning Tasks AMI takes the selected real Tasks and duplicates them using the API to their similarly-named “Planning” Pipeline Step. Links to Assets are maintained for resource analysis, while personnel assignments that would trigger notifications are removed on the Planning Task copies to prevent spam emails. Department planners can then adjust the Planning Tasks completely independently of the real Tasks.

This system supports multiple Planning scenarios by using Shotgun’s Tags. One set of Planning Tasks can be Tagged as “8-hour-days”, another set as “crazy 24/7 proposal”, another as “just use glue and popsicle sticks” etc. Shotgun’s native Filter Panel can be used to switch between the scenarios, by just single-selecting a different scenario Tag.

sg_planning_tasks supports a sync-back mode, where it takes selected Planning Tasks and copies just the changes back onto the real Tasks. Any orphaned Tasks detected during the sync-back are set to Omitted status. This is exposed as another Action Menu Item, just calling the same script but with an argument for sync-back mode.

Scheduling

While fabrication departments have been figuring out quantities of builds and developing their bid schedules, the Story and Editorial departments have been breaking down the Scenes into actual Shots. Shots are generally defined as a set of images to be captured from the same camera angle, notwithstanding pans / zooms etc.
Storyboards in Shotgun
Storyboards are hand-drawn still images representing the key beats during each Shot. The Story Department produces Storyboards using Adobe Photoshop and the Foundry’s FLIX software, which Editorial cuts into “animatics” – videos of boards timed to what will be “real” movie time. We have integrated Flix with Shotgun, so that Storyboards are available in either application and are kept in sync.

Shots and Units
Shots in Shotgun are created programmatically by Editorial publishing an animatic of an entire Sequence, i.e. all the Shots that were broken-down from the Scenes in that Sequence form the script.
LAIKA’s movies tend to consist of about 1500 Shots once all Scenes have been broken down. Each Shot has a wide range of activities that have to be performed – set dressing, rigging, lighting, rehearsals, the actual “hero” animation, additional camera passes without any Puppets present for later rotoscoping, multiple exposures for HDR, etc. So, the number of Tasks required to produce Shots is very significantly more than the number of Oneline Tasks. Production Management is responsible for scheduling those Tasks as well as making daily schedule adjustments as necessary to make sure all those Tasks get completed. Doing that boils down to one simple rule:

**Keep the Animators Animating**

The work the animators do is at the heart of our movies and it is extremely painstaking and difficult. The key is to keep anything from delaying their progress on a Shot. While an animator is working on one Shot on a Unit, the Stage Support teams are preparing another Unit for that Animator to move onto – what we call “bounce to” – as soon as they finish that one Shot. Stage prep involves Lighting, Motion Control programming, Set Dressing, Mechanical Rigging and more – and all of these Tasks have to be reviewed and approved before an animator launches on that next Shot on that next Unit. Managing the Asset deliveries and preparing dozens of stage Units each day depends on tightly coordinated involvement of every department.

To support this multi-department collaboration, stop motion movies have traditionally relied on analog Big Boards. Like everything in stop motion, it’s a very tactile, physical process: cards representing tasks for shots for the entire film are pinned on walls or big boards to show the daily shooting schedule on every unit. Such Big Boards drive every aspect of the production; they tell the shops when they have to deliver a set, puppet, prop, or face kit; the stage crew knows what has to be lit, dressed and framed for the animators; and the VFX team knows when they can expect the turnover of plates. Crew can meet in front of the boards independently throughout the day to plan, resolve conflicts and keep the animators on schedule. The system works; it is visual and easy to understand. It’s also public, centrally located and encourages collaboration between departments.
But the system doesn’t scale well.

**Figure 20.** 6 weeks of *Kubo’s* shooting schedule on Big Boards

**Figure 21.** Close Quarters

One 4’ by 8’ board fits 3 weeks’ of Stage activities on about 2-5 units. A full paper analog Big Boards schedule for a 90-week production would stretch nearly the length of a football field! As our films got more complex, with most shots requiring multiple passes and visual effects elements, it started to get a bit insane. At points on *Boxtrolls* and *Kubo*, we had boards stretching out of the production office, and down hallways, around corners. Making daily adjustments became an overnight task.
Starting with *ParaNorman* (film 2) Laika had begun representing high-level “groups” of Stage Tasks in Shotgun with very coarse resolution by just applying a standard 4-Task TaskTemplate for every Shot. But with 1500 Shots, even just 4-Tasks-per-Shot meant 6000 Stage Tasks to wrangle.

![BigBoards Stage Tasks in Shotgun](image)

**Figure 22. BigBoards Stage Tasks in Shotgun**

Worse, such coarse resolution didn’t even begin to capture the intricacies of activities that were actually being scheduled and revised daily down to 2-hour resolution. Nor did it account for the specific Task requirements of individual Shots as illustrated by their Storyboards and animatics. The 4-Task template may have been fine for a short Shot of a single piece of Kubo’s origami fluttering down, but not for a literally 20-foot-tall skeleton monster forming from flying bones and rearing up with fire coming out of his eyes. The reality is that an accurate Stage schedule is close to 30,000 Tasks for one LAIKA movie.

This all meant that the coarse generic Stage Tasks were not an effective link to the Asset production schedules. Without automatic correlation between fabrication and stage shooting tasks, deliveries were only accurate when tacked-up on the boards. Shotgun’s computer-desktop web-based front-end just wasn’t suitable for the volume of daily data entry and edits required to keep the Shots schedule current and linked to the Asset and crew schedules, not for 6000 Tasks and certainly not 30,000.

#ProTip! At these volumes of Tasks, it’s good to turn on Shotgun’s BigDataMode.
So, the production teams demanded a digital system that could capture the real details of Stage Tasks while being fast, intuitive and capable of understanding blocks of related sub-Tasks. And it would have to maintain the tactile and visual method of building the schedule on boards, as well as collaborative interactions around them. We met this challenge and scaled-up our Shotgun usage during the second half of shooting *Kubo* by building three new systems: **BBGun, The Planner, and The Display**.

**BBGun**

BBGun is a Django-REST-framework service back-end, named after an earlier iPad-based prototype. Its development began long before the introduction of Shotgun’s REST API, so it had to rely on Shotgun’s Python API. It provides datetime math functions that override Shotgun’s, as well as application-specific query and update endpoints. Variable expiry caching is applied to the Shotgun-based endpoints via memcache. Related microservices provide access to Shotgun schedules and media without token authentication, also relying on memcache to reduce response times.

This strategy allows us to send and receive JSON over HTTP thus permitting all manner of clients to safely access and modify the schedule data. This service has opened the door to more custom web app development and gives us the ability to tailor tools for very specific use cases while keeping everyone working on the same verified dataset.

Shotgun’s Task datetime problem can’t be overstated. This is the single biggest limitation of Shotgun as far as LAIKA is concerned, and compensating for it is one of the most important functions of BBGun. The Start and End (Due) fields on Shotgun Tasks currently only support full-day values; they cannot store a starting time, even though Task Durations can be scheduled down to single-minute resolution. But our Stage Tasks absolutely require being able to be scheduled to start at one of four time slots: 9am, 11am, 2pm, or 4pm. We’ve worked around Shotgun’s limitation by adding a `sg_start_time` field on Tasks for BBGun, which handles all the date and time math to recalculate the correct end-date for, say, a 2½ day-long Task starting at 4pm on a Wednesday. Shotgun would calculate such a Task to end on Friday, when in fact it ends 11am on the following Monday. The problem gets worse when holidays and studio production breaks have to be considered or weekend-work has to be scheduled. Ultimately this means that the End dates of Stage Tasks as calculated by Shotgun are often incorrect by at least a day, and sometimes even by weeks. BBGun recalculates the correct end date and time, so dates and times are accurate on our applications such as the Planner and Display that use BBGun as a Shotgun intermediary. We’re fortunate that fabrication Departments are more concerned with Start-dates, and that VFX is accommodating of the fuzziness of the expected turnover end date/time of the shooting Stage Task for a Shot. We’re also fortunate that the other Departments have so far not required start-time scheduling, and so all the existing automation and apps we’ve written work properly with Shotgun’s existing date-math and WorkSchedule rules system. But this problem is only going to get worse and more complicated as we move into our next film, for which Production Management is going to require more fine-grained control to account for individual crew member work schedules and ad-hoc Saturday work, and the fabrication Departments are starting to consider scheduling Tasks at hourly resolution as well. Native time support in Shotgun Tasks remains a significant open concern.
The Planner
The Planner system combines a 55” table-top touch-screen display with a Scala-in-Java native application front-end that communicates to BBGun instead of directly to Shotgun.

Stage Task schedulers needed easy access to several types of operations at the same time, so we divided the UI into three sections. At the bottom is a collection of palettes from which the scheduler can select Shots, Puppets, Sets, animators and stage crew, and Template Tasks, all stored in Shotgun. In the middle, an editing area we call the Sequencer where one combines the resources and Shots to lay them out in the desired shooting order, similar to a Non-Linear Editor UI. At the top is the live schedule to where the scheduler can drag the assembled Tasks to a Stage Unit and publish them to Shotgun.

Changes on Tasks to Assets, dates and durations, and crew assignments can be made in The Planner, with all changes “staged” in the app on local disk waiting to be published back to Shotgun when desired. Additionally, “scenario” data, broad strokes task sequencing or proposed schedule changes, can be published to a dedicated database owned by BBGun. Scenarios can then be lifted from the service database into the live Shotgun schedule at a later time once the user is happy with the new schedule plan.

Schedule editing in the Planner app happens in two modes. Pass cards can be dropped into the Sequencer, arranged, saved, and shared as a pre-planning step. This mirrors the work that would normally happen on a table in the old production alley. Each Pass
Card represents a whole set of Tasks, focused on a single animation shooting Task and including all its contributing sub-Tasks. The Card can include a variety of types of task, pre-determined based on the type of shoot, such as a lightning tweak, set dressing, a rehearsal, and many others.

When rescheduling happens, we often a need to create or close a gap by pushing or pulling a large selection of tasks. This was a labor-intensive process on the old analog boards, but with the digital system it’s just a few taps with the “nudge” tool.

Thoughts for future work on this system includes the ability to handle more flexible per-animator working schedules and some form of “auto-leveling” functionality, to optimize the distribution of activities for time-constraints, or resource-constraints, or release-date. Right now, schedule optimization is still almost exclusively a human function, though the large and custom-designed interface makes it far easier to visualize.

The Display
The Display system combines multiple short-throw projectors with Hoylu’s steganographic surface and Bluetooth imaging pens to provide wireless UI. It presents the wall-sized Big Boards form factor, familiar to all Departments in the studio, while adding enhanced single-user and group interaction capabilities.
The key to the Display system was maintaining the visual language of the cards from the original Big Boards. The color and shape of each task card was maintained in the digital version making the new boards instantly readable. As tasks are published from The Planner, messages detailing the changes are generated by Shotgun and echoed to RabbitMQ. By leveraging RabbitMQ, the Display app listens to the schedule changes via STOMP (a websocket-style connection), allowing multiple Displays to live-update across the studio in near real-time. This ensures that the schedule you’re looking at is always current regardless of how long you’ve had that browser tab open. The Display app is designed to work on desktops and tablets as well, making the schedule available everywhere. We even go so far as to embed it back inside Shotgun in a URL Widget.

The digital Display app allows significant new interactivity. Crew members can drill into a single Task for details specific to their role or zoom out and see the entire shooting schedule for the movie. We even have an alternate “timeless” scale called Cards View, where Shot Task-sets are packed finish-to-start with the same width regardless of their starts or durations, which is particularly helpful for viewing animator bounces. Users can filter for a particular Asset or other resource, view a Shot’s animatic, and leave text notes for other departments. Users can even put free-form hand-drawn notes or sketches anywhere on the digital board, just like on the analog boards.

**Figure 25. Display App Interactive Features**
Preserving the tactile, visual workflows in the Planner and Display apps was worth it – the scheduling team already intuitively knew how to use both the apps. Automatic-updating was a huge win – there are now live Digital BigBoards in multiple meeting areas and conference rooms. Another win was enhanced collaboration – the accuracy and always-live nature of the shooting schedule created social pressure for the fabrication Departments to keep their information equally as accurate and up-to-date. Until BBGun and the Planner and the Display apps truly connected the Asset and Shot schedules, Production Technology could only bring the Department horses to the Shotgun water but could not force them to drink. Their peers in the other Departments could, and do.
Producing, Pre- and During
We’ve continued building on this architecture to develop similar tactile and visual tools, targeting both desktop-based coordinators as well as the mobile workflows of the stage crew during Production. We’ve also continued to dive deeper into specialized workflows, including those for Shops that are less show-specific and more like studio-wide global services.

Shotgun Pages and Native Features

There are many specialized pages for specific during-production activities, including:

- **The RP Shot Sheet** – to ensure face-kits are Just-In-Time for stage animation
- **AD Shot Approval**, where the Assistant Director can track completed Shots hot off the Stages and schedule them for Editorial and Director reviews
- **VFX BB Weekly Targets** and **VFX Render Ranglin’** – feed VFX Department scheduling based on turn-overs of shooting Passes from the Stages
- Shotgun’s native **Review Notes App** used for all VFX dailies
Figure 29. Tracker Pages – Where are Today’s Puppets and Props?

And for many Shotgun Entities, we’ve written workflow-specific Action Menu Items:

Figure 30. Action Menu Items

Save a Frame from RV

Save from action menu in shotgun (right click or use the More menu)
Stage Support Apps
We have teams such as Set Dressers and camera crews that are always mobile, moving between what is now 90 units on our expanding stages. We’re developing apps to put the info and access they need on their phones and tablets and interactive kiosks, instead of having to return to a desktop. These are all Shotgun-backed and reliant on BBGun or the REST API:

Figure 31. LAIKA-MAP: LIVE-UPDATES FROM TASKS TO LOCATE RESOURCES ON ALL UNITS

Figure 32. END-OF-DAYS: PROVIDE NOTES REGARDING CRITICAL ACTIVITIES DURING THE DAY TO LATE-NIGHT-CREW PREPARING UNITS FOR ANIMATORS THE NEXT MORNING
**Figure 33. Puppet Hospital Notes: Medical Charts for Injured Puppets**

<table>
<thead>
<tr>
<th>Shot</th>
<th>Task</th>
<th>Unit Stands By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange 1</td>
<td>5090.0010</td>
<td>Orange 1 Puppet</td>
</tr>
</tbody>
</table>

**Add a note**

**Figure 34. Hand Tracker: Track Interchangeable Hands and Fingers on an iPod Touch**

**Location: Armature**

- Armature/Hand Wiring
  - Status: Waiting to Start
  - 7/8/19 → 7/10/19
- Casting
  - Status: Waiting to Start
  - 7/29/19 → 7/31/19
- Seaming
  - Status: Waiting to Start
  - 8/7/19 → 8/16/19
- Paint
  - Status: Waiting to Start
  - 8/20/19 → 8/30/19
“Global Shop” Requests and Resources
As LAIKA’s fabrication schedules have become more complex, starting earlier before shooting, they have begun to overlap in-progress productions. This led to the recognition that various Shops were actually not movie-specific. Activities such as producing large-format prints and machining custom parts for rigs and armatures were not necessarily associated with a single production, i.e. a single Shotgun Project. Further, these activities functioned more on a per-request, on-demand basis rather than needing to be scheduled in advance. We needed to provide a way for a person or Department to submit a request to one of these show-agnostic Shops, in response to which that Shop would create a few Tasks of their own as needed.

At first, we considered using Shotgun’s Ticket and Note entities, but both of those entities have the drawbacks that they are designed to be Project-specific and they have Shotgun-customized Detail pages. We wanted a Request entity that could be either global i.e. non-Project-specific, or to be on behalf of multiple Projects. We also wanted complete control over its Detail page. Therefore, we made use of a CustomNonProjectEntity for Requests, and created new custom fields on it to support the various different Shops which this entity would support. We made use of the Type field to distinguish the different Shop Requests, although we could also have used a Group field as well.

A basic example of a Shop Request using a modal AMI is for the Print Shop:

A user from a production Department can visit the Print Shop queue page from Shotgun’s Global Pages menu and make a new Request for a large-format print using a custom Flask-based form as shown at top. New Requests then enter the Print Shop queue, where Shop personnel fulfill them in a Kanban-like fashion.
Machine Shop Pipeline
This Request system has recently opened the door to an entirely new integration with another specialized show-agnostic service organization, LAIKA’s Machine Shop. The Machine Shop is part of the LAIKA’s Rigging Department. Essentially, Rigging is responsible for holding things up. This can take the form of armatures inside puppets, or external rigs holding practical effects that will be rotoscoped-out by VFX, or even larger motorized structures supporting sets or motion-controlled camera rigs.

Since LAIKA’s puppets are all custom-made, their internal armatures consist of custom-machined metal parts. During *Boxtrolls*, we deployed the Request entity-based system for users in the Rigging and other Departments to make Requests for custom parts.
Recently, we focused on the Machine Shop workflow. They use Autodesk Inventor and Vault to design and track the custom parts. But as with Assets and Shops, there wasn’t a good connection between the parts, the fabrication process, and the delivery / needed-by management. The Machine Shop had been using a small database of their own design to keep track of the parts, the processes, all their equipment, and their inventory. We realized all that information and inventory could be tracked more effectively and integrated with the delivery flow in Shotgun. Parts could be correlated with the Assets (Puppets, Sets, etc.) for which they were required, enabling cost management and future-needs projections. Machine Shop personnel can create a TaskTemplate for each Mechanical Part, and whenever a Part gets Requested, just applying the TaskTemplate can create the required Tasks complete with machine- and process-specific settings.

So, we’re deploying a suite of new CustomNonProjectEntities and also plugins for Inventory, Vault, and FeatureCAM to read and write them. This project had the fortune of starting just as Shotgun’s REST API beta became available, so these custom plugins did not need to rely on a Python interpreter e.g.IronPython just to enable the Shotgun python API. The REST API is used exclusively. This was especially important because the vast majority of LAIKA’s production processes are Mac and Linux based and thus our standard pipeline toolset does not provide significant Windows support, whereas the Machine Shop is virtually all Windows-based. The REST API helps bridge that divide.

**FIGURE 39. MECHANICAL WORK CENTERS**
Figure 40. Mechanical Parts

Figure 41. Mechanical Tools

Figure 42. Machine Shop Workflow Pipeline
**Figure 43. Vault <> Shotgun Sync**

**Figure 44. FeatureCAM to Shotgun**
Tracking
Great! So, we’ve scoped and bid a movie, and are producing it while adjusting its schedule. The final requirement to tie it all together is to measure the progress of the production. Deadlines and milestones must be achieved to hit the movie’s release date, which is negotiated separately with distribution partners and theaters and other stakeholders.

Shotgun supports displaying statuses and percentages at any single point in time quite well, but only recently has begun introducing native functionality to graph data over time. And while graphs can be useful, LAIKA needed to be able to measure progress on many different fronts, with many different metrics, right from our first movie. During Kubo, we fleshed-out our use of Shotgun’s WorkSchedule system and designed another entity specifically for tracking data over time. But first, we needed to ensure Shotgun understood our calendar.

WorkSchedule and Schedule Memcache
Shotgun’s WorkSchedule system allows setting a standard working schedule “globally” for the whole studio (site), with overrides per-Project, which can in turn be overridden per-Person. As Shotgun administrators, we ensure that weekends, holidays, and scheduled production breaks are entered into Shotgun so that Task end-dates get calculated as accurately as they can be (the datetime problem discussed in the BBGun section notwithstanding). Further, some Departments are maintaining employee schedules as well in Shotgun, again so that Task scheduling reflects resource availability.

We have found that API calls to the WorkSchedules can be slower than desired when significant changes are being pushed to Tasks. To resolve that, we’ve implemented an automated job that runs before the studio opens every morning, that stores WorkSchedule data in various fast-access python data structures in memcache. For increased performance, applications that need to access Shotgun work-schedule data during the day first check the memcache, and only make API work-schedule calls in the event of a cache miss. Significant new WorkSchedule EventLogEntry events during the day also trigger a memcache refresh.

Week Entity
Now with our studio and personnel calendars being maintained in Shotgun, we can connect progress to time. One of the primary consumers of the WorkSchedule is our progress-tracking entity, a CustomEntity (project-specific) that we named a Week.

Any production process that needs to measure progress over time is going to do so on some periodic basis, and so an entity that represents that period can itself be defined and configured in Shotgun. We happen to measure progress on a weekly basis. Other production processes might be more rapid, needing a Day entity, or slower which might suggest a Month entity. Regardless of the measurement unit, the concept works.

When a new Project gets underway at LAIKA, we populate its per-Project table of Weeks. Each Week has a Start Date (a Monday) and a custom name including the Production Week number; we number principal photography Production weeks starting...
at 1, and therefore pre-Production weeks have descending negative numbers. Weeks also have a Status value flagging them as pre-Production, or principal Production, or post-Production which are Weeks occurring after the last week of principal photography. A quota_from_oneline script populates Planned target “quota” values for all the planned Production shooting weeks. Automation using the same WorkSchedule library creates-new and updates-existing Weeks as needed, such as in response to major rescheduling events like a new shooting-start date or release date.

The bulk of the Weeks table is then made up of Shotgun Query Fields performing specialized per-Department and global-level Queries on the values and statuses of other Shotgun entities linked to those Weeks. The key is to link the Assets being produced to the Tasks producing them, and then to link the periodic entity in which key events occur to those Tasks as well. We created three critical new fields on Tasks: Start Week, Weeks, and End Week. These fields get populated and refreshed constantly using the shotgun_event_daemon. When the start date of a Task changes, the Production Week in which that start date occurs is stamped into the Start Week field. Similarly when the end date of a Task is changed, the Production Week of that new end date gets set into the Task’s End Week field. Either type of change in turn causes a calculation to find all the Production Weeks encompassed by the Start and End Weeks inclusive, and that list gets set in the Task’s Weeks field.

![Figure 45. WEEK ENTITY SCHEMA AND A QUERY FIELD CONFIGURATION](image-url)
Show Progress

In that way it becomes possible to perform native Shotgun queries for Tasks that start, or end, or occur-within any one particular calendar Week, returning that Week entity instead of just a date range. Having that Week entity complete with all its Query Fields, shown in a List view sorted by the Weeks’ Monday start date, provides a comprehensive yet concise summarized data view of key progress metrics over time.

It doesn’t end there. Other critical production events can be stamped onto the production entities as well. For example, we use Shotgun’s Version entity to represent the final still images and videos that ultimately form the movie; when a Version gets reviewed and approved to appear in the movie, we’ve written automation to stamp the “Week Approved” corresponding to that status-change event date onto the Version. We do the same sort of automation for entire Shots, with the same sort of “Week Approved” field. Editorial sets the “cut duration” for each Shots – that is, the number of still images that make up the shot. LAIKA produces our movies at 24 frames-per-second – and so, we can sum-up the progress of the number of movie-time-seconds being produced on a Weekly basis using a Query Field for the cut-durations of Shots, filtered by Shots whose Week-Approved is each Production Week, divided by 24. Accounting can designate certain Weeks as Milestones against which to track progress.

The Week entity has proven useful in other ways as well. The daily WorkSchedule memcache automated script also refreshes the number-of-working-days in each Production Week on each active Project, ultimately allowing us to account-for the
percentage of actual calendar working-time that any given Task encompasses, since Start and End and Duration values do not intrinsically reflect weekends and holidays and human assignee work-schedule exceptions like days-off. The Shotgun reverse-fields created by having the Start Week and End Week and (encompassed) Weeks fields on the Task entity lets us look at any one particular Week entity and easily see what Tasks are scheduled to start, or finish, or are active-within that Week, and what percentage of the days in that week are actually working days. Query Fields at the top-level Project entity can query for the number of pre-Production or Production or post-Production weeks along with overall-Project metrics summarized from those Weeks.

We call these views of weeks “Show Progress” pages. We can combine the Weeks values on Tasks by filtering on the Department-specific Pipeline Steps and thus create filtered views of Week-Progress reports on per-Department bases.

**FIGURE 47. WEEKS, DETAIL VIEWS**
Usage Reports

Finally, since Tasks have Assignees – that is, people assigned to perform them – and Tasks are associated with Weeks, we can easily construct dynamic and live Shotgun pages that detail personnel usage per Assignees’ home Shops over time.

**Figure 48. Usage Report: Assigned Headcounts per Shop per Week**
Conclusion

Factory Parallels
There are significant similarities between LAIKA’s Shotgun-based movie production processes and any traditional factory environment. The workloads for a new initiative are scoped and bid to generate cost estimates and optimize resource usage. Schedules are generated from high-level estimates then iteratively refined and detailed to provide Just-In-Time deliveries to downstream departments. Facility-wide service organizations perform production support activities including order administration and inventory operations. Production progress is measured on an appropriate periodic basis that permits timely reaction and management.

Automation
Shotgun has the capacity to handle very significant data loads, with very esoteric data models and business logic. At large scales it must be supplemented with significant automation to ease the user-interface and data entry requirements and maintain interdependent relationships. The Python and REST APIs, AMQP support, and reactive event daemon system provide solid bases for developing workflow automation. Much like many Digital Content Creation applications, the full power of Shotgun is often only unleashed by facility-specific customizations and automation. Developers must work in conjunction with users to fully understand the workflows and data models, to develop automation that builds trust in the system and engenders user adoption.

Collaboration
However, no matter how centralized, detailed and accurate the data model and collection, it cannot replace human interaction. Shotgun has been an effective tool for us to make everything that’s happening during our productions visible and thus facilitate collaboration, but it cannot make the decisions for us.

As we near the release of our fifth production, the forthcoming Missing Link, we’ve noticed an important side-effect of our digital success. All the tools have made information faster to enter and easier to share, but they cannot force people to comprehend and react to it. The previous model of analog paper big boards necessitated not only in-person gathering but also a slower update pace, which encouraged collaborative decision-making. During Missing Link, we’ve found at times that the efficacy of the digital tools and automation led to schedule changes being so easy and fast that taking the time for collaborative decision-making was neglected.

Don’t mistake data integrity for a hive mind, nor let it replace discussing changes and decisions. The tools can make data more accurate and more available to make better-informed decisions, but those decisions still need to be discussed and communicated effectively.

MISSING LINK
IN THEATERS
APRIL 12TH 2019
**Bonus Content!**

These topics will not be covered in detail in the presentation.

**LAIKA’s Shotgun Arsenal**

We locally host all our services. LAIKA doesn’t currently have facilities in multiple geographies.

**IT Services**

- PostgreSQL cluster: load-balanced and proxied
- Memcache provided by a portion of our render farm
- AMQP cluster with RabbitMQ broker
- NFS of course
- virtual machines for Shotgun nodes

**Shotgun Nodes**

- 5 production: redundant, nginx load-balanced
- 2 staging: redundant, load-balanced, test version just before production
- 1 R & D: single node, experiment with newest version to pick for staging
- 3 old-show servers: one each for *ParaNorman, Boxtrolls,* and *Kubo*
  - the single centralized production “server” is a better approach
- 1 partially exposed to the internet through a DMZ for client users
  - special access and legal approved materials only, not for tracking
- 1 cloud-hosted-by-Shotgun: also for R & D but without any Intellectual Property

**Automation**

- Jenkins for periodic and triggered jobs, 10 VM’s (4 Mac, 6 linux)
  - do-our-own Version thumbnail transcoding, using our render farm
  - “accumulation” math – cumulative sums of values over serialized time
  - “auditors” – hourly checks for correctly-linked Tasks and resources
- ~50 shotgun_event_daemon engine plugins for “instantaneous” events
  - Production and testing

**Microservices in Docker containers**

- Mainline and PubHub
  - event/message aggregation, notifications and report subscription services
- sg-widgets
  - Flask+Jinja backend for embedded-in-Shotgun custom forms and reports
- shotgun.pt
  - thumbnails without tokens nor authentication, memcache’d for speed
  - shortcut URLs: instead of /detail/Asset/8675309, write /kbo/Asset/Monkey
- BBGun backend
  - schedule Task dates with time support, canned-query endpoints
- Metrics with the ELK Stack (*Elastic Search + Logstash + Kibana*)
- Error reporting with Sentry

**Applications We’ve Written**

- Over 200 Action Menu Items for Shotgun and RV
  - and 2 custom browser protocols to support them
- Plugins for DCC packages like Maya, Houdini, Nuke, Hiero, and more
- Standalone apps for image and editorial publishing; no Shotgun Desktop. 😎
An Op-Ed about Permission Roles and Rules

I recommend keeping Shotgun’s Permission Roles and Rules as simple as possible. Remember that Shotgun records history in the form of EventLogEntries. History Logs are your best friend because they expose everything, which helps everyone see what’s really going on, thus establishing trust.

We rely on the accuracy and completeness of the history logs with our production departments, because their response to an un-trustable or incomplete history leads to demands to “implement lots of permission rules, and snapshot backups and reporting, and more notifications.” As Shotgun admins we get vast mileage out of being able to say, “don’t worry so much about permission rules and micro-managing what can happen within your Shotgun workflows, because there are complete records of everything that happens. So we can detect and undo anything that goes awry.” When that falls apart and trust is broken, lots of unnecessary make-work happens. So even if something bad like a producer’s budget or a department’s delivery schedule got changed, it can be tracked to who did what and when, then undone.

And an unexpected change “allowed” by open permission rules can be a good thing! Uncovering what’s going on in your system is not only an auditing aspect, it can reveal new unknown evolving workflows about which you may have been previously unaware. Try to avoid creating unnecessary frustration in your system. This applies to any tool or system, not just Shotgun. When someone without the “right” permission rules tries to do something in your system but is denied – even though they really may have a valid business need to do so! – they’ll stop using or trusting your system. While some users will ask for permissions to be fixed, too many will defer to someone else and think “this system is not designed for me.”

Security can be a valid, even paramount use for Permission Rules of course. But business logic can often be more flexible. Be careful not to conflate business logic with security – that leads to abuse of Permission Rules and ultimately much more administration work than you may have the technical staff or interest to support.

Security can be a valid, even paramount use for Permission Rules of course. But business logic can often be more flexible. Be careful not to conflate business logic with security – that leads to abuse of Permission Rules and ultimately much more administration work than you may have the technical staff or interest to support.

Consider also if you’re thinking of using a Permission Rule just to enforce a workflow behavior. Perhaps you want to guarantee a certain naming convention for Assets of a particular Type. You could consider not allowing users to Create such entities, and instead require them to use an Action Menu Item that would launch a custom application of your own design that implements your naming-convention restrictions. You can then take advantage of a variety of Shotgun features to still reflect the user that created the entity in the EventLogHistory:

- the Shotgun API allows writing the created_by field during the initial .create() call, or
- connect with a script key (APIUser) but use the sudo_as_login feature to substitute the user’s login in subsequent EventLogEntries, or
- “embed” the user’s login from an environment variable such as $USER into EventLogEntry UUID’s with the .set_session_uuid() method, with an AMI to retrieve and decode that UUID later as needed for auditing.
Encoding and Decoding UUID’s in Shotgun

Sample code in Python:

```python
>>> import os
>>> import uuid
>>> user = os.environ['USER'] # taiello
>>> user_uuid = str(uuid.UUID('{':<16}'.format(user)[:16].encode('hex'))
>>> user_uuid
'74616965-6c6c-6f20-2020-202020202020'

# now, assuming you have already established a Shotgun Python
# API connection using shotgun_api3.shotgun.Shotgun, named sg:
>>> sg.set_session_uuid(user_uuid)

# now do anything that generates an EventLogEntry

>>> event = sg.find_one('EventLogEntry', [], fields=['session_uuid'], order=[{'field_name': 'id', 'direction': 'desc'}])
>>> user_uuid = event['session_uuid']
>>> user_uuid.replace('-', '').decode('hex').strip().decode('ascii')
'taiello'
```
References and Suggested Reading

“LAIKA’s digital big boards”  [https://dl.acm.org/citation.cfm?id=3085034](https://dl.acm.org/citation.cfm?id=3085034)

“Shotgun MS-Project Plugin”  [https://github.com/taiello/sgProjectPlugin](https://github.com/taiello/sgProjectPlugin)


“Anoto Creative Presents the FLIX Wall”  [https://www.youtube.com/watch?v=2BA8gvXJV4w](https://www.youtube.com/watch?v=2BA8gvXJV4w)

“Shotgun Event Framework”  [https://github.com/shotgunsoftware/shotgunEvents](https://github.com/shotgunsoftware/shotgunEvents)


“Semantic UI React”  [https://react.semantic-ui.com/](https://react.semantic-ui.com/)