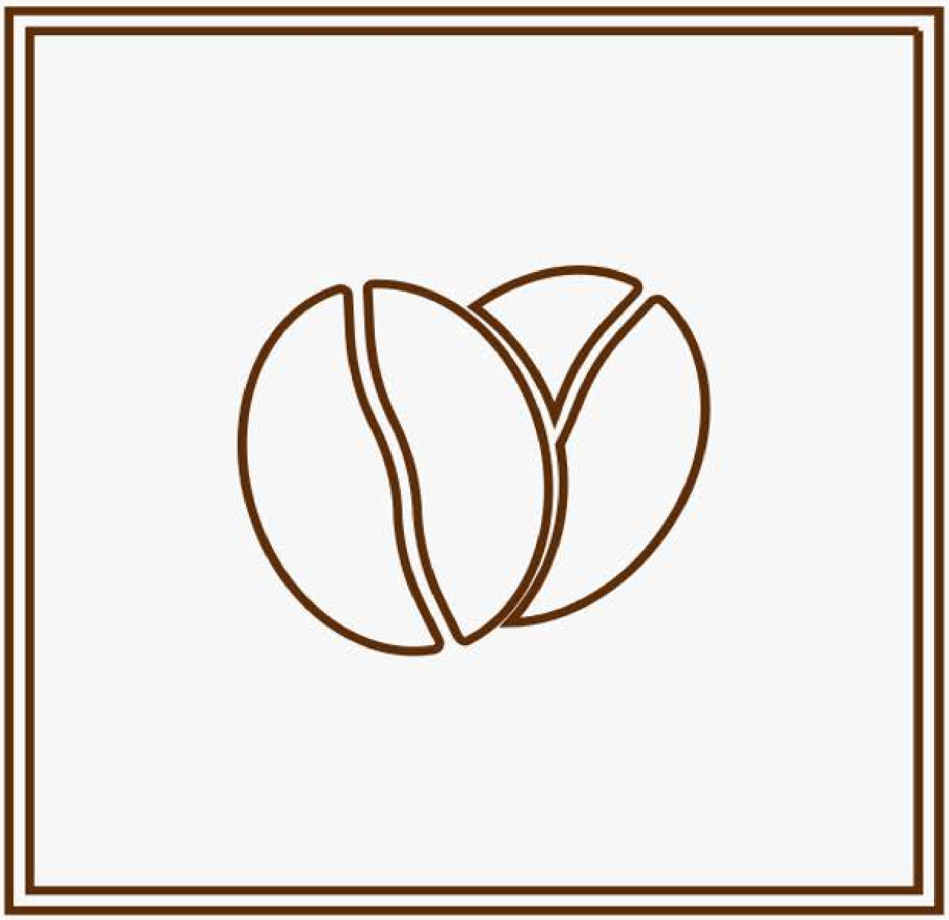


APPLYING DOMAIN DRIVEN DESIGN WITH CQRS AND EVENT SOURCING

BY NICK CHAMBERLAIN

Chapter 1

The Domain

First Pop Coffee Company is a business specializing in roast-to-order coffee. *Roast-to-order*, in this case, means that the company has scheduled roast days where he roasts a set number of lbs of coffee based solely on the supply and the amount that he can roast in that single roast session. He doesn't have the resources to roast and sell large quantities of roasted coffee beans, but he wants to grow and scale as demand increases.

The company is run by Nick Chamberlain who has been a hobby roaster for 5 years. He started with a sample drum roaster that can roast 1/4 lb per batch. He recently upgraded to a 1 lb per batch drum roaster. Each roast takes about 10 minutes from start to finish. With prep time and warm-up time, he can roast 13 lbs per hour with his current setup. That means that on a given roast day, he can roast about 104 lbs of coffee. He buys it bulk from a supplier and they sell 50 or 100 lb bags via UPS (former) or freight (latter).

A local roaster is selling their "select" coffees for $15.00 - $18.00 per lb. If he can buy 100 lbs of coffee for around $500 - $550 and sell the roasted coffee for $15.00/lb, he could generate about $1560 revenue per roast session. Minus bean cost and overhead, he's thinking he could do about $2000 - $2500 profit on a good week.

While he has the potential to make more if he can buy more equipment and roast more per session, he doesn't want to sell more than he can produce. Hence his idea that he sells only the amount that he can roast at any given time. So he hired a software company to develop a website for him that will let him be really transparent to his potential customers about the timing of his roast-to-order coffee.

What does this mean? Well, Nick is open to suggestions. He is imagining having a way to schedule his roasts by calendar day. So on his website, visitors can see what days he has scheduled to roast. Based on his current supply of coffee and the age of the many shipments of green coffee he has in stock (ready to be roasted) he will decide what days he has available to roast. His calendar might be a 1 week calendar or a 1 month calendar, not quite sure yet.

Say he has 200 lbs of green coffee in stock. He wants to roast 5/7 days this week. He will plan his roast days for Monday thru Friday. When he updates the calendar through a management console, he can allocate as many pounds that he wants to roast for any of his roast days, along with what kind of bean, cupping notes, and ideal roast based on samples that he's done (he won't do custom roast levels). He might plan to roast 30 lbs each day (the default schedule), but he can deviate. He can also do half of the roasts as one bean and half as another. He'll have the ability to allocate certain beans for each batch.

When he sets up his roast schedule, the coffee roasted that day will become available for purchase from the website. He will choose the bean that he has available and it will be displayed on the schedule with cupping notes.

The user will click on a day and then will see the 30 lbs (or scheduled amount) that are available to buy for that roast day. They might be represented graphically, like a grid that they can click squares to add a lb to their cart. They will see other customers reserve lbs for the day and when all the lbs for the day are purchased, no more lbs can be purchased.

The cart system might work like an event ticket reservation system. You may add the lb to the cart, but to make sure the customer commits - it will be removed from the cart after 5 minutes. Only after the customer charges their credit card will the coffee be reserved completely and no one else will be able to buy it.

Once a customer purchases coffee from a roast day, they will get a confirmation email with the schedule and the expected dates that the coffee will be roasted and shipped to them.

On roast day, Nick will confirm that the roast day has started and the user might get a notification that their coffee is being roasted today. After roasting the coffee and bagging it, Nick will confirm that he roasted all the lbs of coffee that were reserved for the day and a notification will go to the user that their coffee was roasted. The roast schedule on the website will reflect the day's status - scheduled, roasting, roasted, resting (1 day rest period), shipped.

When the coffee gets shipped after the 1 day rest period, the customer will get notification that their order is being shipped. Shipping will be a flat rate box of some sort (hoping there's a service that can take care of it). The coffee will get bagged in a plastic zipper bag with CO2 vent.

He'll need to know what he has available when he sets up his schedule. So when he receives bulk coffee from his supplier, he'll need to be able to add the green beans to inventory and that will be reflected in his admin console that he uses to allocate batches to the schedule.

The site will also sell roasted beans (dated) when available at a lower price. Maybe ~$10.00 for coffee that was roasted the earlier week. This will be a separate page on the website for discount coffee that's not roast-to-order.

There will be a mailing list that will notify people about the next schedule release.

He's not quite ready to approach it yet, but he plans to also offer custom K-Cups at 14g of coffee each cup so the math could be done to figure out whether that'd be profitable.

He's also thinking about giving the customer a copy of their roast profile as a marketing thing.

His competitive advantage comes from offering the freshest artisan-roasted coffee available and giving himself the freedom to control demand through his scheduling system. If he wants to roast less, he's able to schedule less. If he wants to make some more money, he can roast extra for the week (given that he has enough customers and demand)...

Big Picture EventStorming

The Team helping Nick get his website going begins with an EventStorming session. This is so that both the software Team and Nick can get their heads wrapped around the Core Domain and make sure they are writing the right code. It's essential that Nick uses 3rd party for the code the Team shouldn't be writing because he has a limited budget.

This is EventStorming session is called the Big Picture (Brandolini, 214) event and it will be the first of a few EventStorming sessions while they develop the Ubiquitous Language and Context Map for the system.

I won't define a lot of terms for DDD or EventStorming. I want the reader to do the homework for term definitions. I'll be using capitalization to do 2 things:

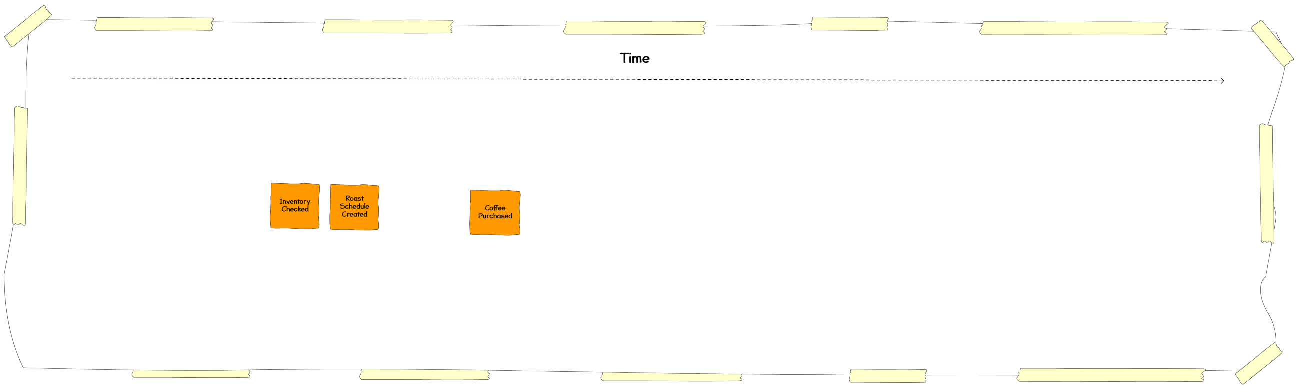
- Emphasize important Domain-Driven Design terms that you can research on your own
- Emphasize important Ubiquitous Language terms that emerge through this discussion

I want to focus on end goals. At this point, we're trying to achieve a working Context Map that we can prototype quickly so that Nick can evaluate whether the Team is on the right track with his business plan.

Our first Domain Event: "Coffee Purchased"

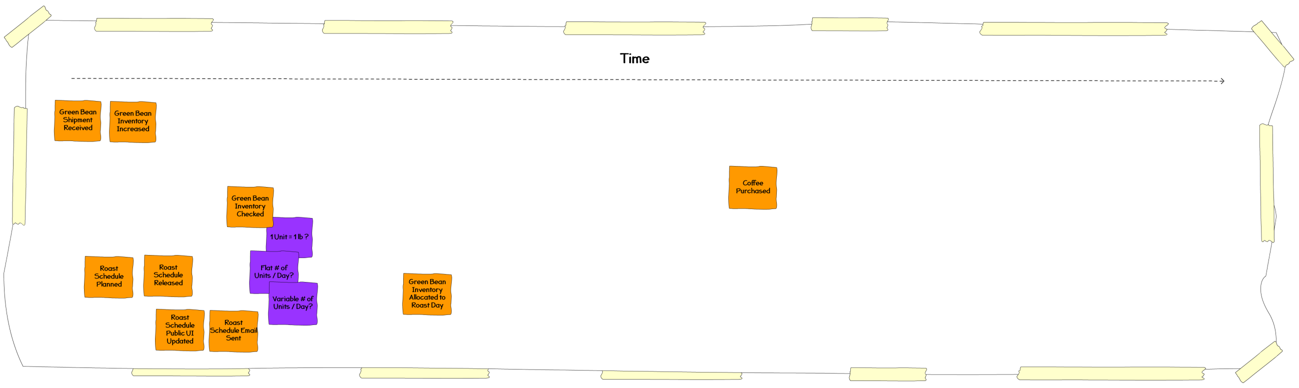
The Team asks Nick to think of an event that will happen in the core of his business. He wants to be very customer-focused, so he writes **Coffee Purchased** on an orange stickie and slaps it up on the board. This is what he's hoping for the most, obviously.

What else is he envisioning as core to his business? There's the idea of his "Roast Schedule" so he puts **Roast Schedule Created** as another Domain Event. In order for him to make good decisions while he plans his Roast Schedule, he'll have to monitor his Inventory of green, un-roasted coffee beans. So the next Domain Event: **Inventory Checked**.



Momentum

Minus a couple periods of silence, broken by the Team putting a random Domain Event up on the wall (a technique used by EventStormers coined "Icebreaker"), the Team continues to fill in a timeline of Events across Nick's roast-to-order business domain.


Time
Green Bean Shipment Received
Green Bean Inventory Increased
Green Bean Inventory Checked
1 Unit = 1 lb ?
Flat # of Units / Day?
Variable # of Units / Day?
Roast Schedule Planned
Roast Schedule Released
Roast Schedule Public UI Updated
Roast Schedule Email Sent
Green Bean Inventory Allocated to Roast Day
Coffee Purchased

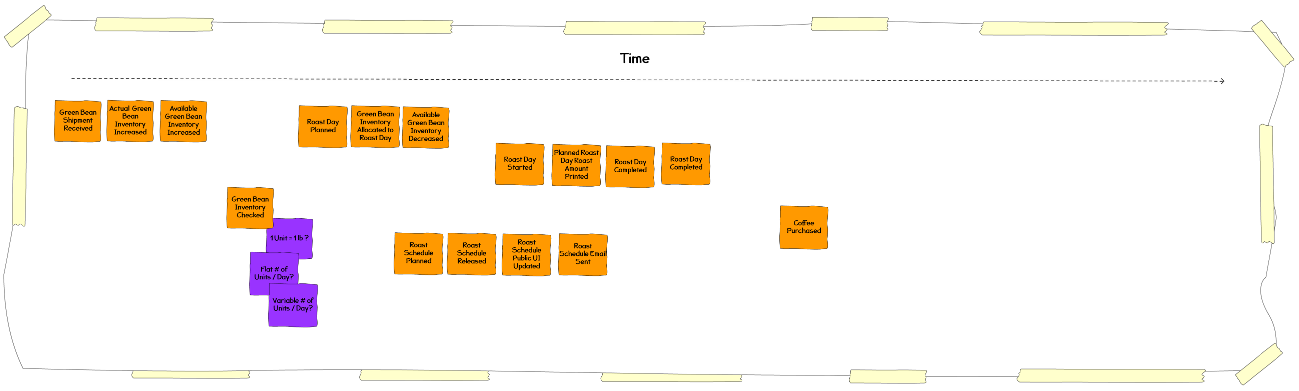
As Nick and the Team start putting more Domain Events on the EventStorming surface, questions or "Hot Spots" start to emerge:

- Is the primary unit of measure 1 lb of coffee?
- Is Nick going to always roast a fixed # of lbs of coffee per day?
- Or will Nick be able to roast more coffee one day and less on another?

We always note them on purple stickies and slap them on the wall near their respective Domain Events.

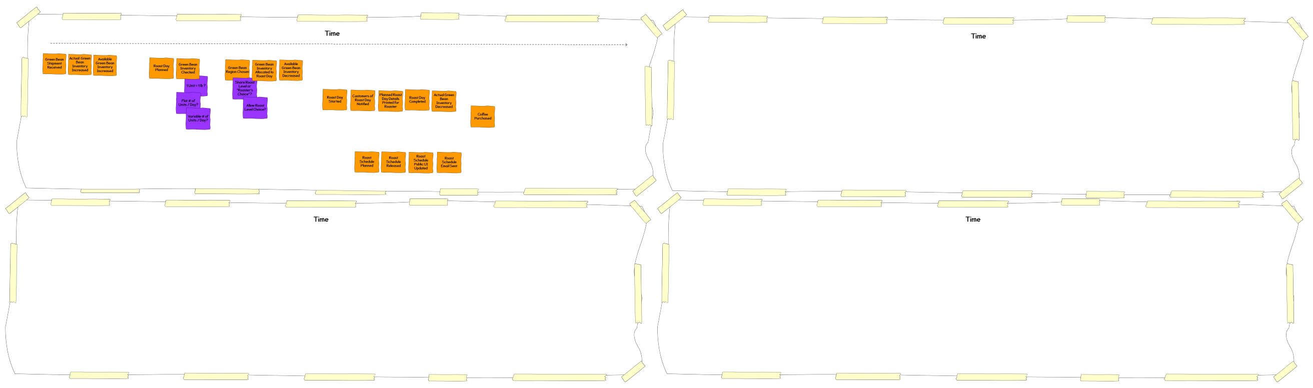
Terms that Nick defines as a Domain Expert start to emerge:

- Green Bean: Coffee that hasn't been roasted yet. Purchased in bulk, either 50 or 100 lb bags.
- Roast Schedule: The outlay of Nick's production week where he plans to roast x lbs of green beans each day for a selection of days in the week.
- Inventory: When Nick receives a shipment of green beans, they get placed in Inventory and available to allocate to his Roast Schedule.


Time
Green Bean Shipment Received
Actual Green Bean Inventory Increased
Available Green Bean Inventory Increased
Roast Day Planned
Green Bean Inventory Allocated to Roast Day
Available Green Bean Inventory Decreased
Roast Day Started
Planned Roast Day Roast Amount Printed
Roast Day Completed
Roast Day Completed
Green Bean Inventory Checked
1 Unit = 1 lb ?
Flat # of Units / Day?
Variable # of Units / Day?
Roast Schedule Planned
Roast Schedule Released
Roast Schedule Public UI Updated
Roast Schedule Email Sent
Coffee Purchased

Unlimited space

Whether or not we find ourselves running out of space, we want to avoid even the perception that there is a limited space on our Modeling Surface (Brandolini, 189). As soon as the Team sees the potential for running out of room on their initial paper, they add as much additional space as they can...

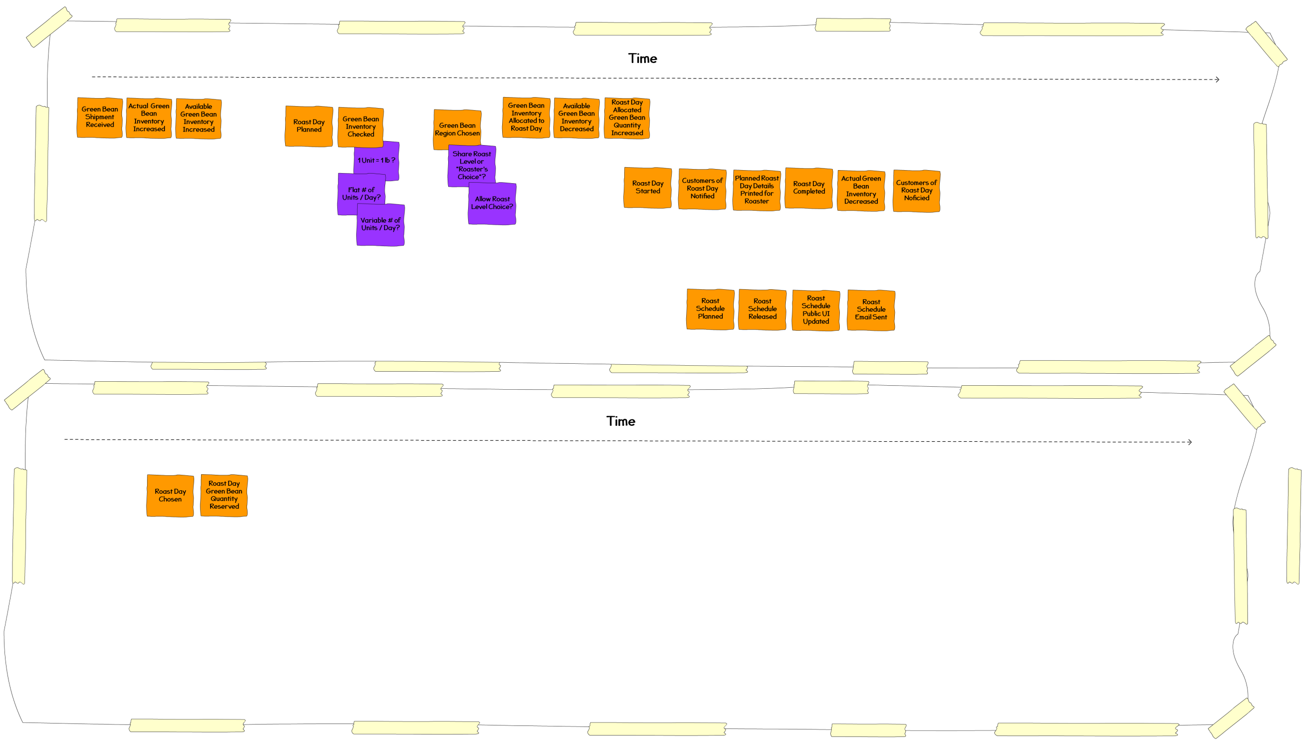

Time
Green Bean Shipment Received
Actual Green Bean Inventory Increased
Available Green Bean Inventory Increased
Roast Day Planned
Green Bean Inventory Checked
1 Unit = 1 lb ?
Flat # of Units / Day?
Variable # of Units / Day?
Green Bean Region Chosen
Share Roast Level or "Roaster's Choice"?
Allow Roast Level Choice?
Green Bean Inventory Allocated to Roast Day
Available Green Bean Inventory Decreased
Roast Day Started
Customers of Roast Day Notified
Planned Roast Day Details Printed for Roaster
Roast Day Completed
Actual Green Bean Inventory Decreased
Coffee Purchased
Roast Schedule Planned
Roast Schedule Released
Roast Schedule Public UI Updated
Roast Schedule Email Sent
Time
Time
Time

Discovering the Domain through EventStorming

Once Nick describes the important Domain Events that he envisions for managing his own Inventory and Roast Schedule concepts, we start to focus on the Customer Experience of buying roasted coffee from him.

Nick and the Team imagine the Roast Schedule as a centerpiece of the site; the first thing that the Existing Customer/Potential Customer sees when they access firstpopcoffee.com.

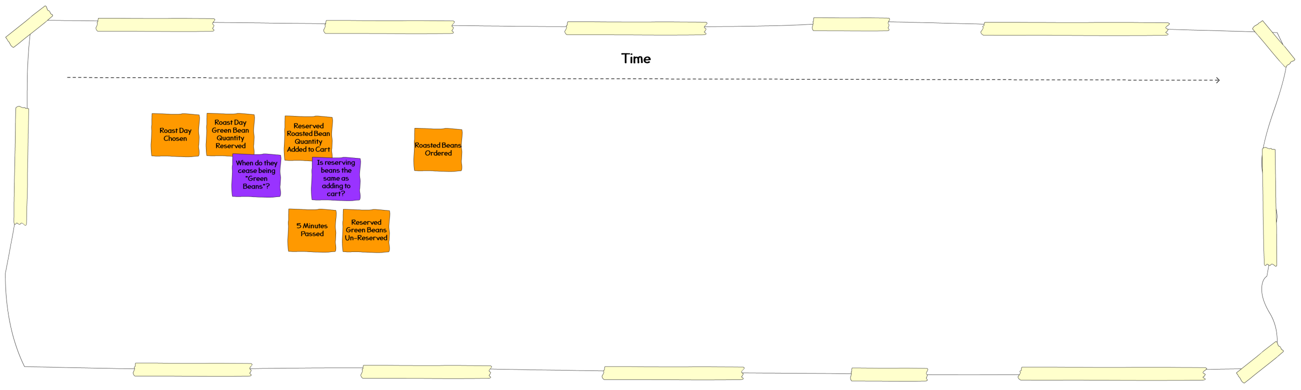
If the Customer is interested in buying Coffee (I'm capitalizing key Ubiquitous Language terms also), they will likely choose a Roast Day from Nick's Roast Schedule. The Domain Event is **Roast Day Chosen**.



Time
Green Bean Shipment Received
Actual Green Bean Inventory Increased
Available Green Bean Inventory Increased
Roast Day Planned
Green Bean Inventory Checked
1 Unit = 1 lb ?
Flat # of Units / Day?
Variable # of Units / Day?
Green Bean Region Chosen
Share Roast Level or "Roaster's Choice"?
Allow Roast Level Choice?
Green Bean Inventory Allocated to Roast Day
Available Green Bean Inventory Decreased
Roast Day Allocated Green Bean Quantity Increased
Roast Day Started
Customers of Roast Day Notified
Planned Roast Day Details Printed for Roaster
Roast Day Completed
Actual Green Bean Inventory Decreased
Customers of Roast Day Noficied
Roast Schedule Planned
Roast Schedule Released
Roast Schedule Public UI Updated
Roast Schedule Email Sent
Time
Roast Day Chosen
Roast Day Green Bean Quantity Reserved

What makes this domain more complex than a simple "Add a Product to a Cart and Checkout" is that we are "Auctioning Off" a Product that will have a finite Available Quantity allocated to a single Roast Day. It'll be similar to a Venue with limited Available Seating. This means that the potential Customer can't leave Product in their Cart without making a decision within, say 5 minutes. They need to allow other Potential Customers a chance to actually buy the Coffee that's available on that Roast Day.

Nick and the Team start putting new Domain Events up on the wall. Nick is in the mindset that he's going to "roast the green beans according to his available inventory and schedule he designed". So he starts putting Events up like **Roast Day Green Bean Quantity Reserved** and **Reserved Green Beans Un-Reserved**. A purple stickie goes up: "When do 'Green Beans' cease being 'Green Beans' and become 'Roasted Coffee for Sale'?"...


Time
Roast Day Chosen
Roast Day Green Bean Quantity Reserved
When do they cease being "Green Beans"?
Reserved Roasted Bean Quantity Added to Cart
Is reserving beans the same as adding to cart?
5 Minutes Passed
Reserved Green Beans Un-Reserved
Roasted Beans Ordered

This thought process tugs at the core of Domain-Driven Design. Does the Potential Customer really care about Nick's inventory of Green Beans and how he is managing his Roast Schedule outside of simply wanting to buy fresh coffee beans? True, both the Customer and Nick have one focal point - the Roast Schedule. For Nick, he's using the Roast Schedule to plan and control both the demand and supply for his roasted green coffee beans. For the Customer, they're using the Roast Schedule to buy artisan-roasted coffee.

So the question was asked "When do the 'Green Beans' cease being 'Green Beans' and become 'Roasted Coffee for Sale'?"... the answer is that they don't. Customer and Roaster are two separate **Bounded Contexts**.

This is where Domain-Driven Design fixes traditional modeling. What is a core reason why we end up with unmanageable, spaghetti integrations...? We **assume** that a concept that lives in one context **MUST** live in another context.

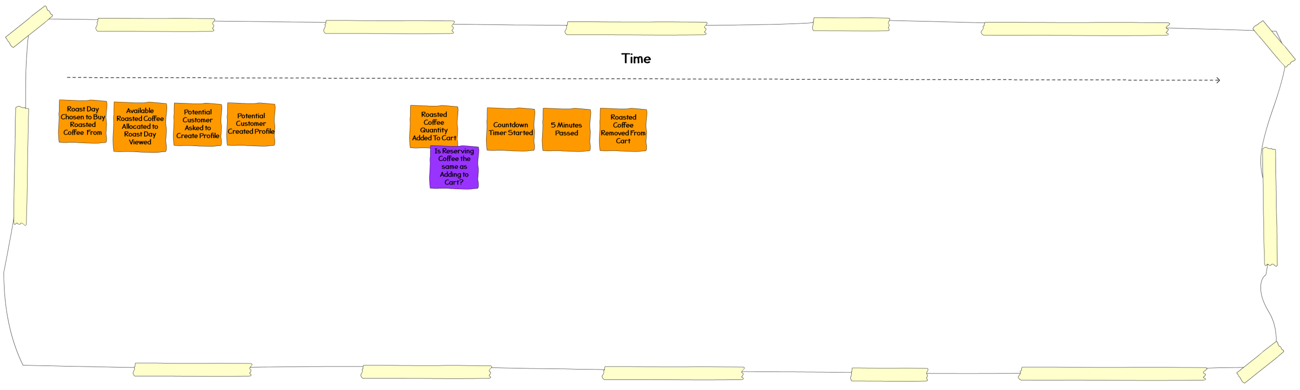
While the Team notices this distinction, they decide to save the delineation of these Bounded Context for a "Design Level" EventStorming session later. They do, however help Nick to establish the different terms in the Ubiquitous Language as he continues putting more Events on the wall.

Revised Ubiquitous Language Glossary:

- Roasted Coffee: Roasted coffee that is purchased from firstpopcoffee.com. It is the end product of roasting the allocated Green Bean Inventory for a given Roast Day
- Green Bean: Coffee that hasn't been roasted yet. Purchased in bulk, either 50 or 100 lb bags.
- Roast Schedule: The outlay of Nick's production week where he plans to roast x lbs of green beans each day for a selection of days in the week.
- Inventory: When Nick receives a shipment of green beans, they get placed in Inventory and available to allocate to his Roast Schedule.

The Whole Picture

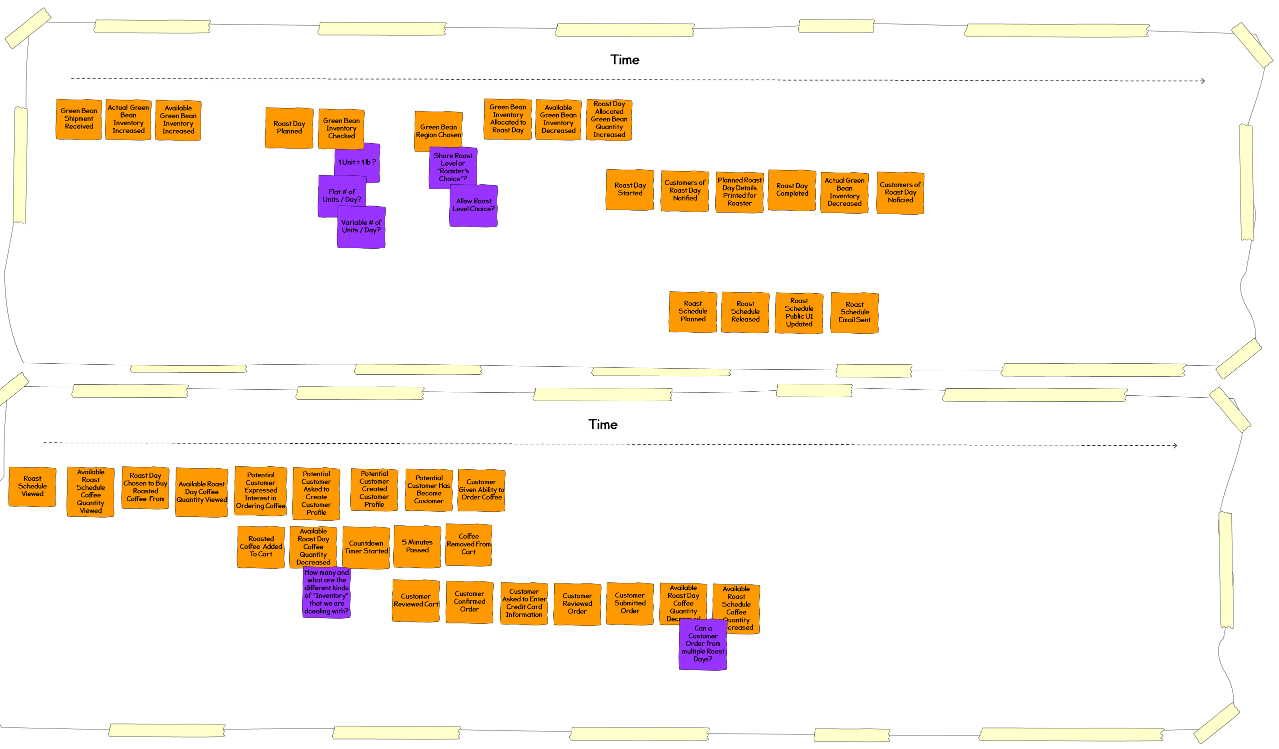
Nick and the Team continues, filling in all of the Domain Events that they can think of and come to agreements about. Questions arise that they have to discuss further like "Is Reserving the Coffee the same as adding the Coffee to your Cart?" After a little discussion, they determine that **Roasted Coffee Quantity Added to Cart** signifies the beginning of the **Countdown Timer Started** Event to give the Customer five minutes to order the Coffee.


Time
Roast Day Chosen to Buy Roasted Coffee From
Available Roasted Coffee Allocated to Roast Day Viewed
Potential Customer Asked to Create Profile
Potential Customer Created Profile
Roasted Coffee Quantity Added To Cart
Is Reserving Coffee the same as Adding to Cart?
Countdown Timer Started
5 Minutes Passed
Roasted Coffee Removed From Cart

We're looking for a Big Picture of Nick's business. This applies for any kind of business you want to use EventStorming for. This first session is an attempt to see everything that's in everyone's head about the business, to break down information "silos."

Discovering Hotspots through EventStorming

As Nick and the Team continue, they discover that within the two Bounded Contexts that have emerged through the exercise - there are various concepts of Inventory. A purple stickie goes up: "What are the different kinds of 'Inventory' we are dealing with?"


Time
Green Bean Shipment Received
Actual Green Bean Inventory Increased
Available Green Bean Inventory Increased
Roast Day Planned
Green Bean Inventory Checked
1 Unit = 1 lb ?
Flat # of Units / Day?
Variable # of Units / Day?
Green Bean Region Chosen
Share Roast Level or "Roaster's Choice"?
Allow Roast Level Choice?
Green Bean Inventory Allocated to Roast Day
Available Green Bean Inventory Decreased
Roast Day Allocated Green Bean Quantity Increased
Roast Day Started
Customers of Roast Day Notified
Planned Roast Day Details Printed for Roaster
Roast Day Completed
Actual Green Bean Inventory Decreased
Customers of Roast Day Noficied
Roast Schedule Planned
Roast Schedule Released
Roast Schedule Public UI Updated
Roast Schedule Email Sent
Time
Roast Schedule Viewed
Available Roast Schedule Coffee Quantity Viewed
Roast Day Chosen to Buy Roasted Coffee From
Available Roast Day Coffee Quantity Viewed
Potential Customer Expressed Interest in Ordering Coffee
Potential Customer Asked to Create Customer Profile
Potential Customer Created Customer Profile
Potential Customer Has Become Customer
Customer Given Ability to Order Coffee
Roasted Coffee Added To Cart
Available Roast Day Coffee Quantity Decreased
How many and what are the different kinds of "Inventory" that we are dcealing with?
Countdown Timer Started
5 Minutes Passed
Coffee Removed From Cart
Customer Reviewed Cart
Customer Confirmed Order
Customer Asked to Enter Credit Card Information
Customer Reviewed Order
Customer Submitted Order
Available Roast Day Coffee Quantity Decreased
Available Roast Schedule Coffee Quantity Increased
Can a Customer Order from multiple Roast Days?

- Physical Green Bean Inventory that Nick is concerned with keeping enough supply through ordering more green beans from his suppliers.
- Allocated/Unallocated Green Bean Inventory that Nick is concerned with for use in his Roast Planning.
- Available Roasted Coffee on a given Roast Day that the Customer is allowed to add to their Cart, either buying the Coffee or after 5 minutes the Coffee goes back to the Available Roasted Coffee for the Roast Day.

Another form of separation starts to show itself between the first two "Inventories" listed above. There is Physical Green Bean Inventory that Nick receives through Shipping and sits in storage, and there's the Allocated Roast Schedule Inventory which is reduced as Customers buy Coffee and he roasts it for them.

The Team, again, starts to see separate Bounded Contexts emerge within the Domain. There's the concept of **Physical Inventory Management** - Nick's Physical Green Bean Inventory that fluctuates as he orders and receives Green Beans into storage. Then there's Nick's **Roast Planning Inventory** - The Green Bean Inventory that Nick can Allocate to his Roast Schedule safely, without scheduling and selling more Green Beans than he can roast.

As they discuss this separation, Nick and the Team agree that there is a separation between what Nick has in storage and what he can allocate and roast. He may allocate Green Beans to his future Roast Schedule, reducing his Unallocated Physical Green Bean Inventory, but his **Physical Inventory** stays the same.

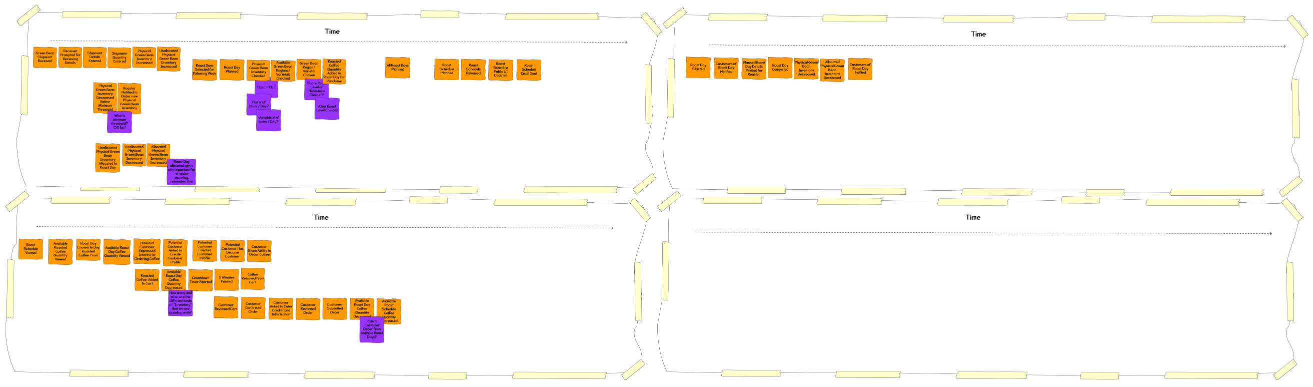
These new terms in the Ubiquitous Language can be incorporated in the EventStorming session:

- Physical Green Bean Inventory: Green bean inventory that Nick orders from green bean suppliers in bulk and stores. He needs to manage what he has in stock in order to make sure he has enough to allocate to his Roast Schedule

- Allocated Roast Schedule Inventory: Green beans that Nick plans to roast according to his Roast Schedule. It is the amount of green beans that he plans to roast as he plans his future roast schedule. It influences (but is not part of the same Bounded Context as) the amount of Coffee that a Customer can buy.

In addition to these concepts of Inventory in the new Bounded Contexts (**Physical Inventory Management** and **Roast Planning**) that are emerging, Nick and Team can also address the concept of **Available Coffee for Sale** in the **Customer** Bounded Context that emerged above. They start calling it the **Customer** Bounded Context, however there is a likelihood that it will be further broken down into finer-grained Bounded Contexts as they move forward.

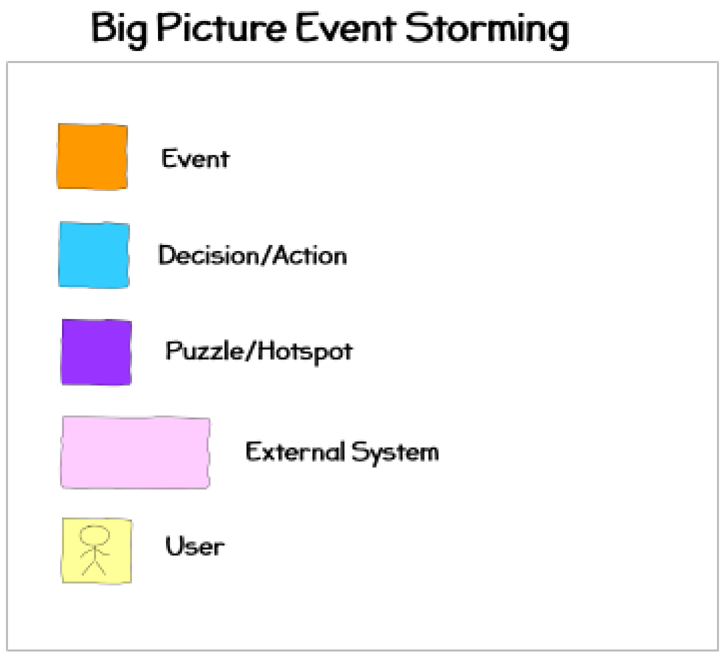
More Events go up on the surface, more Hotspots, and here's where we are:



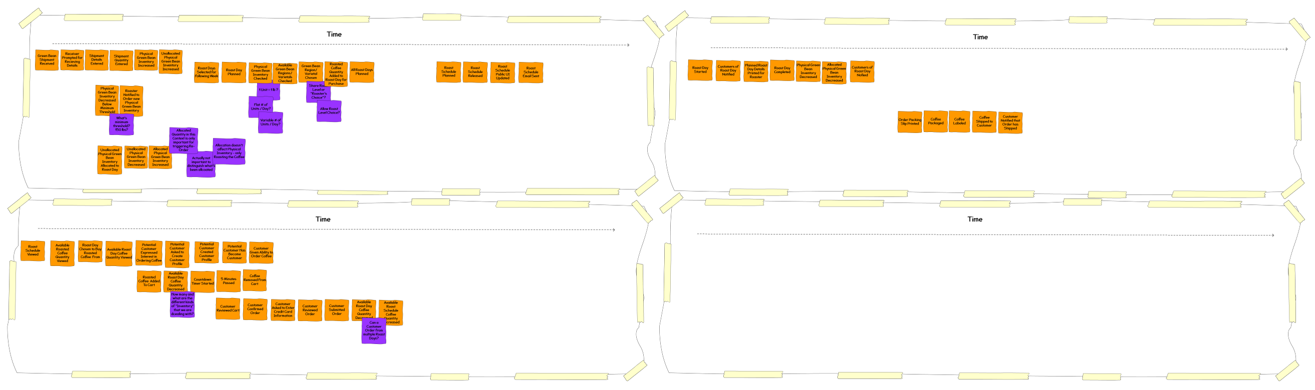
Discovering Commands through EventStorming

Nick and the Team have put up enough Domain Events on the Big Picture Surface to start thinking about the Commands that cause these Events. As you might expect, having sufficient space is going to be important since many of the Events are triggered by some type of Command (some by other Events).

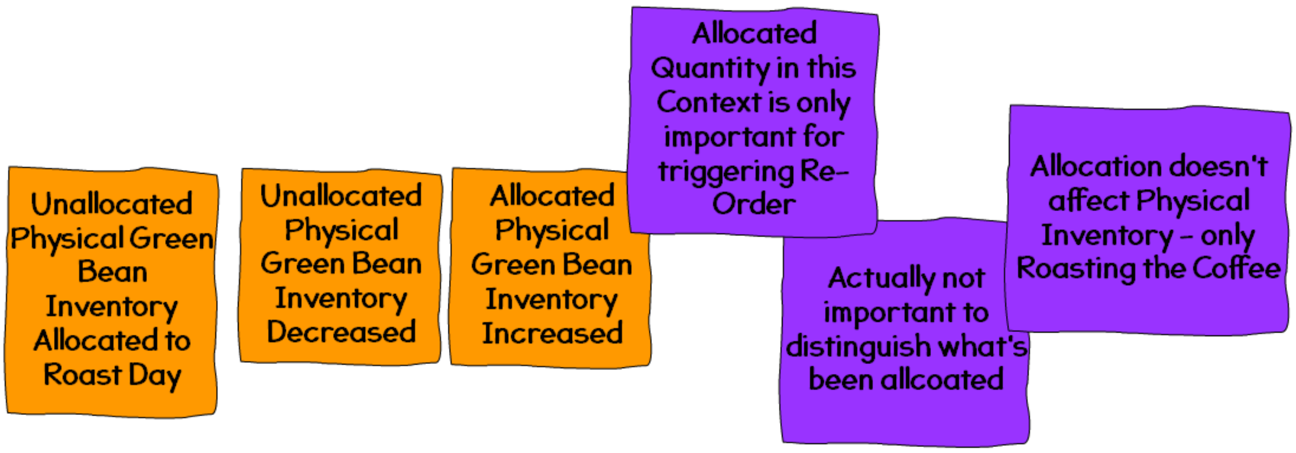
Here is a legend representing the additional stickies going up on the surface:



As a starting point reference, here's what the Surface looks like.



The Team is still kind of fuzzy on the concept of Physical Inventory and Allocated Roasted Coffee Inventory, but they note it as a purple stickie ("Hotspot").


Unallocated Physical Green Bean Inventory Allocated to Roast Day
Unallocated Physical Green Bean Inventory Decreased
Allocated Physical Green Bean Inventory Increased
Allocated Quantity in this Context is only important for triggering Re-Order
Actually not important to distinguish what's been allcoated
Allocation doesn't affect Physical Inventory - only Roasting the Coffee

The momentum picks up as Nick and the Team add Commands to the design surface, keeping an eye on any insights that they have missed while focusing on Domain Events.

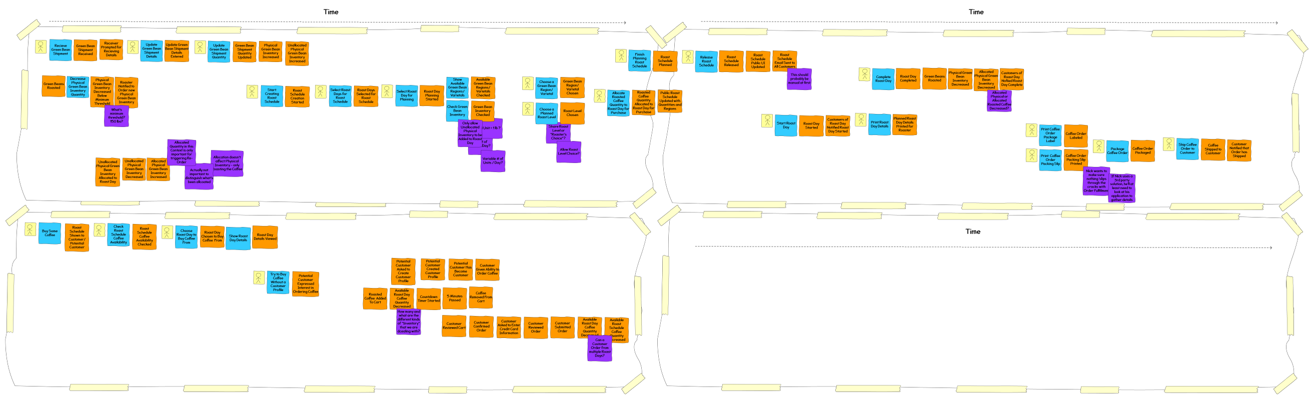
One concept emerges that no one had thought about. The Team suggested that Nick not focus on a custom application for Shipping. This is not the Context that contributes a lot of competitive advantage to his business and can be accomplished by 3rd party tools like Shopify (more on this later).

However, Nick does not want to be bogged down with the overhead of going through every Order when he's done Roasting to make sure he's followed the right steps to Ship the Roasted Coffee to the Customer. He's hoping to have a way to at least go down his Roast Day Itinerary and make sure he's processed every order perfectly. He also needs to think about Dissatisfied Customers when something doesn't go right.

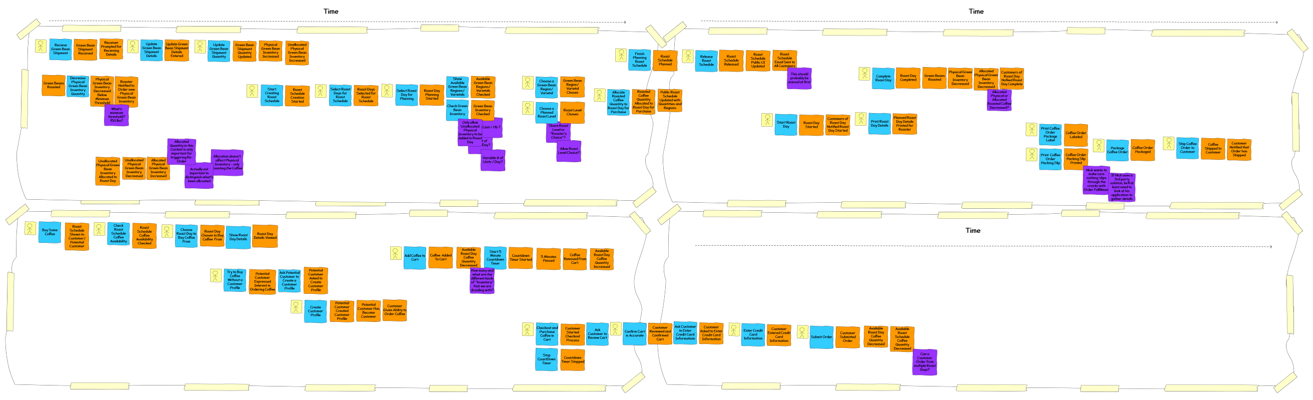
Whether or not Nick and Team agree on using a 3rd party tool for Shipping and Customer Service Bounded Contexts, Nick's "sanity checking" Order Fulfillment Bounded Context may contribute to Nick's competitive advantage enough to consider designing it from scratch. The Team takes note with a purple stickie and moves on.

After lengthy discussion, breaks, and an additional session, Nick and the Team have a Big Picture "Artifact" (Brandolini, 136) to work with. The following figures show the progression:

Time
Time
Time
Time



Here's what Nick and the Team end up with at the end of their Big Picture EventStorming session(s):


Time
Time
Time