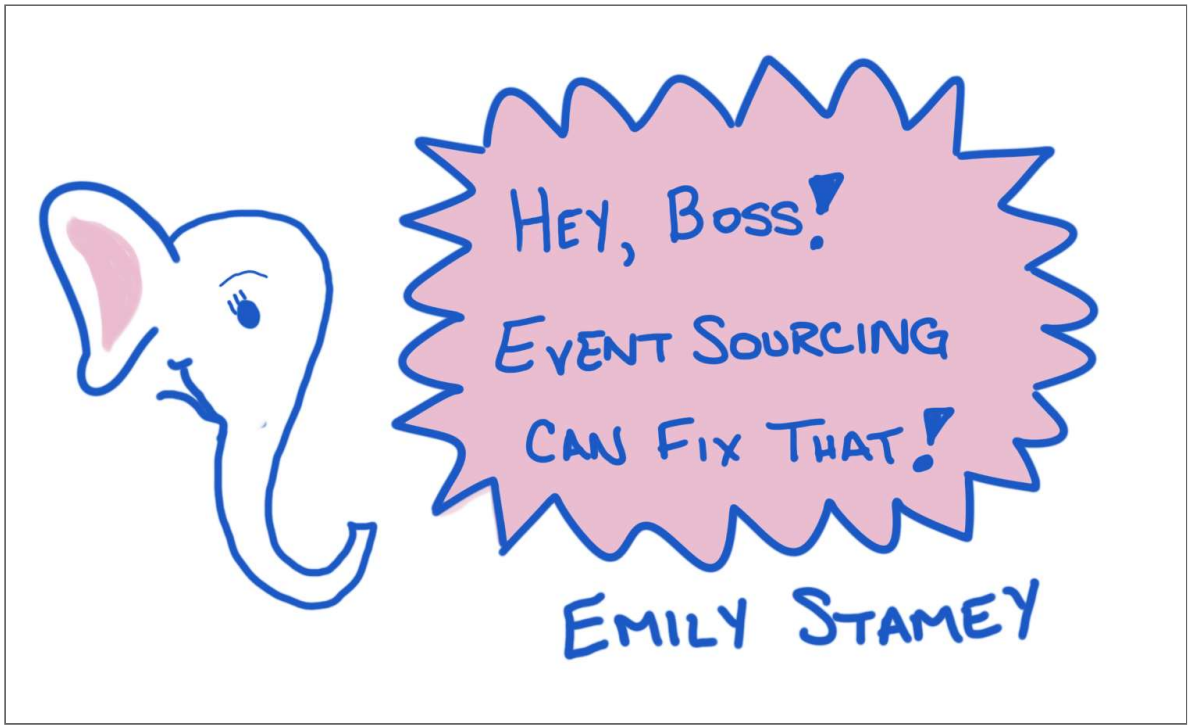
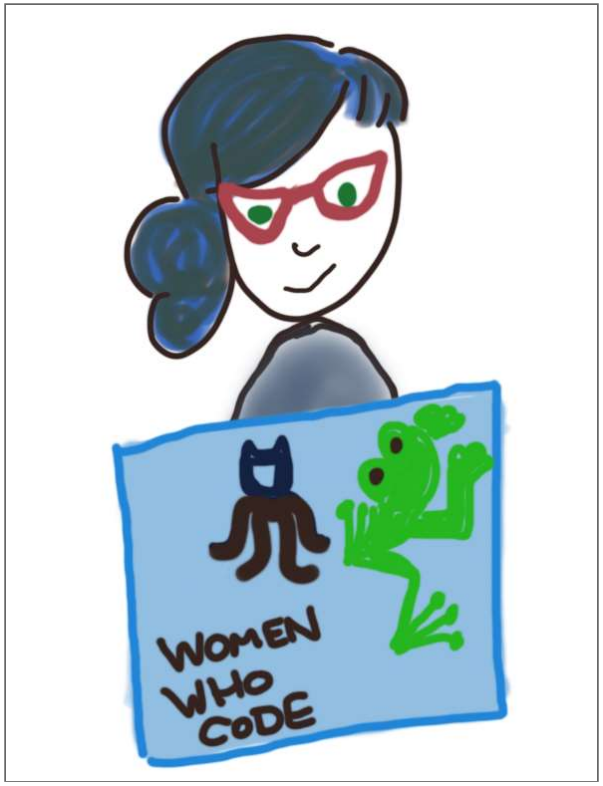

HEY, BOSS!
EVENT SOURCING
CAN FIX THAT!
EMILY STAMEY

DRAGO MI JE STO SAM OVDE



WOMEN WHO CODE RALEIGH/DURHAM

TRIANGLE PHP USER GROUP

INQUEST

BEFORE WE BEGIN

Switch from CRUD to Event-Sourcing is tough

Event-Sourcing isn't for your whole application

LEARNING OBJECTIVES:

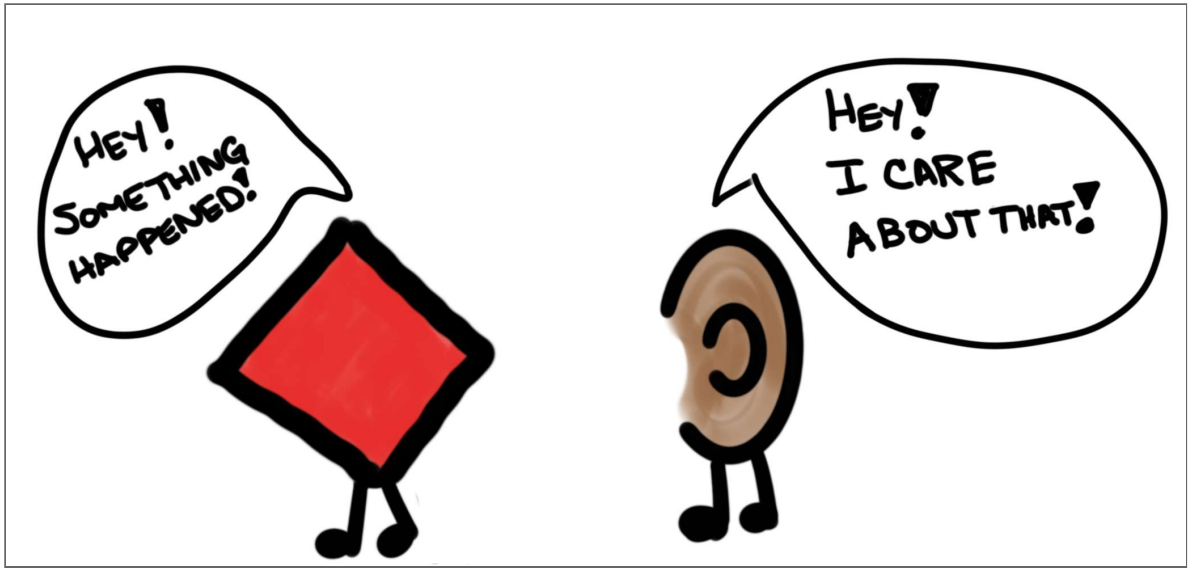
- Basics of Event Sourcing
- Adding Event Sourcing to existing projects and the problems they can resolve

EVENT SOURCING

The fundamental idea of Event Sourcing is that of ensuring **every change to the state** of an application **is captured in an event object**, and that these event objects are themselves stored in the sequence they were applied for the same lifetime as the application state itself.

Martin Fowler

EVENTS AND LISTENERS



AN EVENT

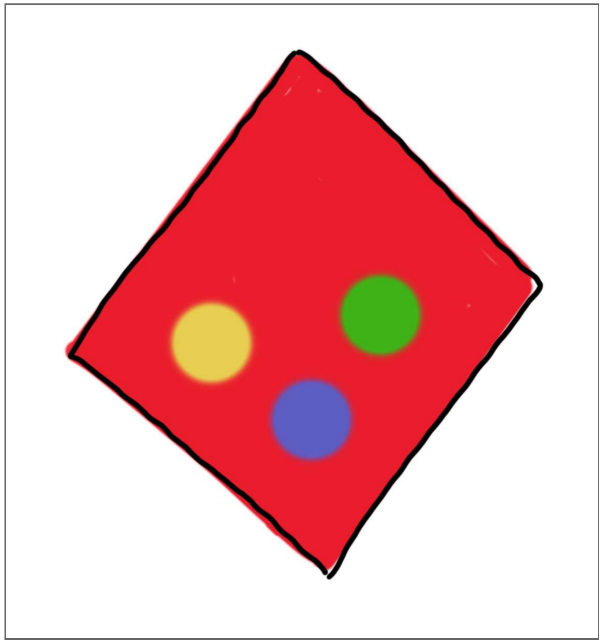
What happened?
BookWasCheckedOut

What do I need to remember about it?
(book, patron, date)

EVENT ATTRIBUTES

- Save only what you need to preserve
- The rest can be looked up

(book id, patron id, date)



EVENT CLASS

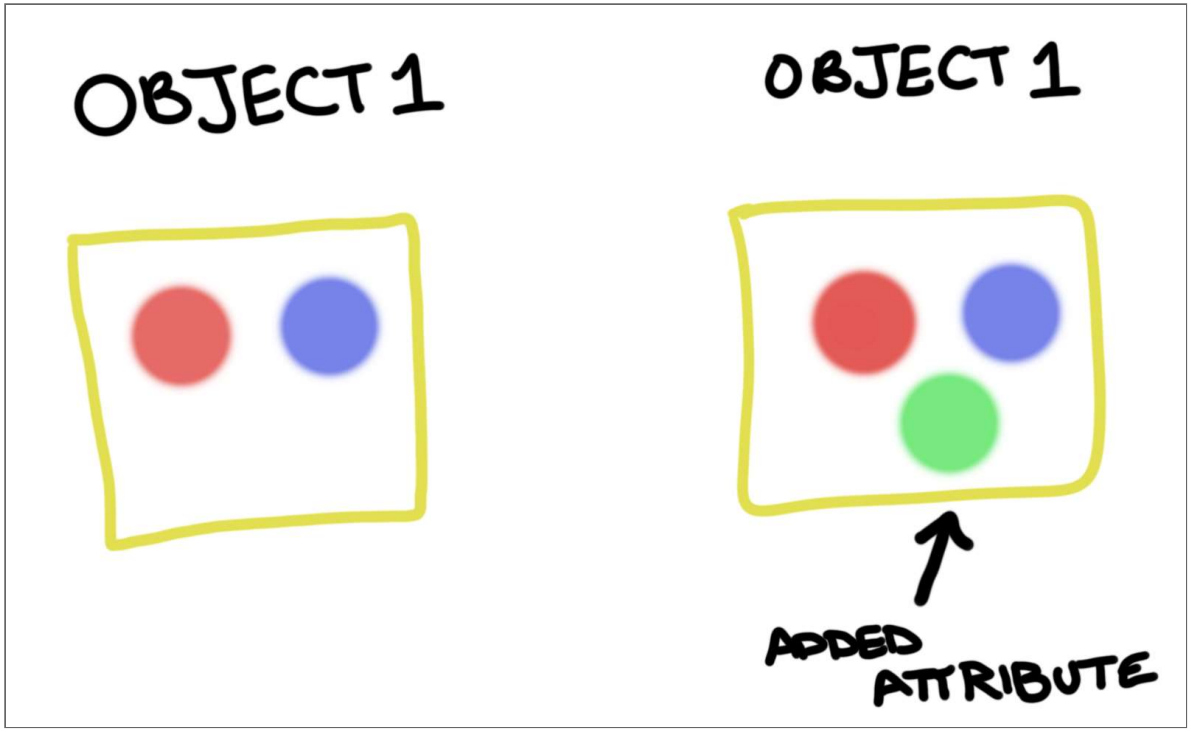
```php
<?php

namespace Library\Events;

use Library\Support\Event;

class BookWasCheckedOut
{

    /**
     * @var DateTime
     */
    public $checkoutDate;

    /**
     * @var int
     */
    public $patronId;

    /**
     * @var int
     */
```

RULES FOR EVENTS

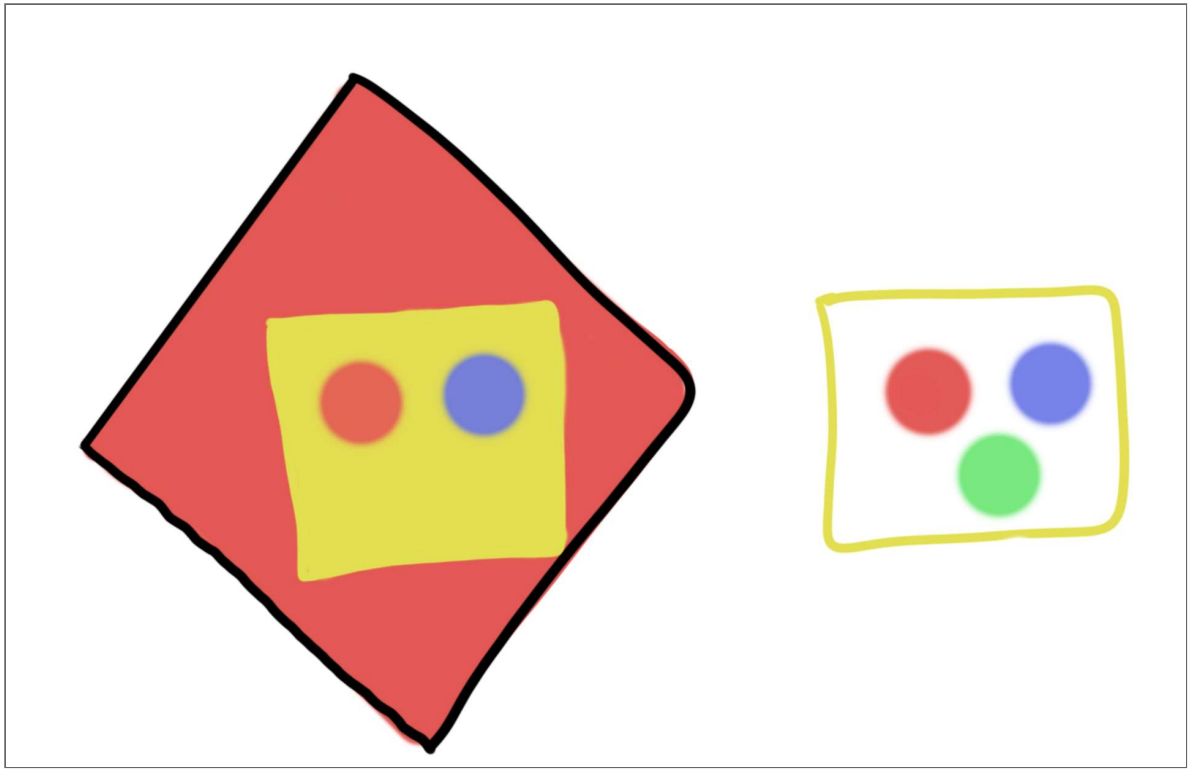
- Usually named as past-tense verbs
- **RARELY** changed
- **Never** deleted
- Has attributes that are values
  - not model, object, collection, or aggregate root


NEVER DELETE EVENTS
DEPOSIT $500
CORRECTION -$500
DEPOSIT $60

DON'T STORE OBJECTS



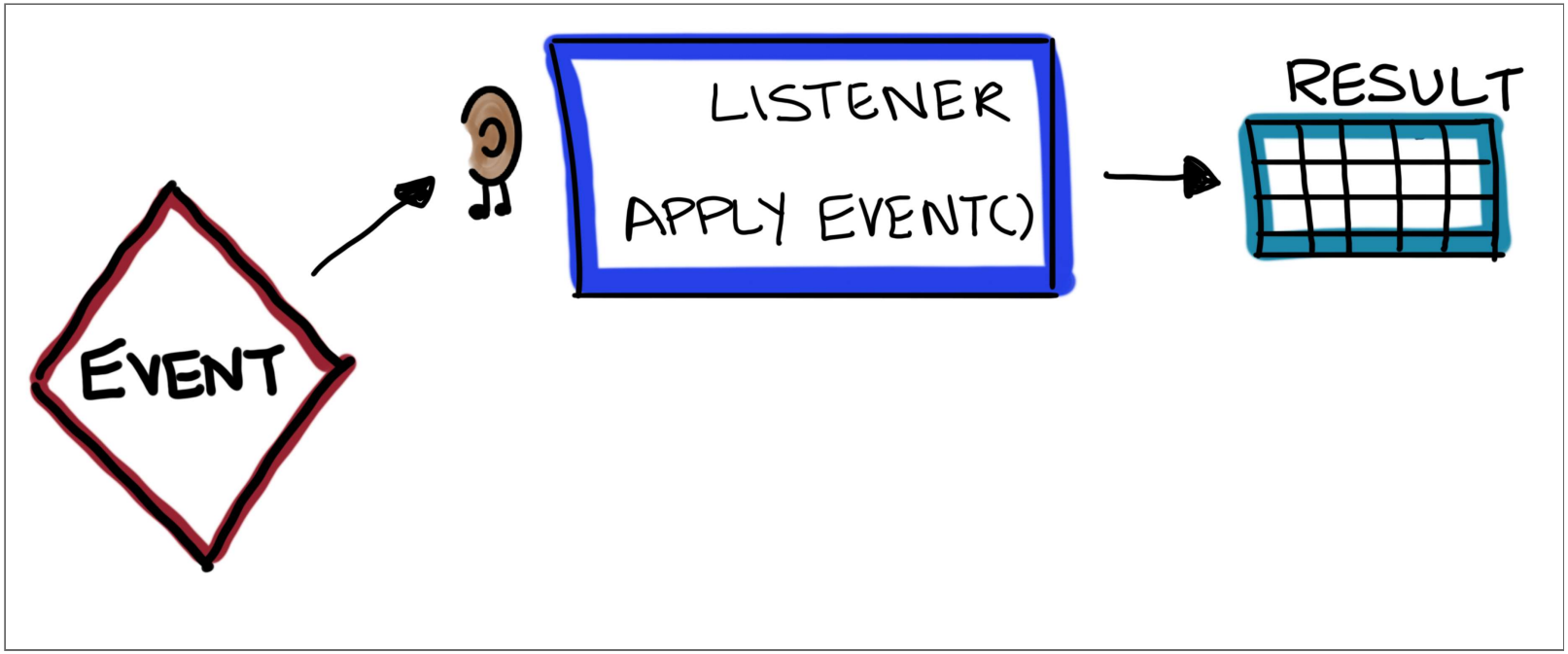
IF WE STORED OBJECTS IN AN EVENT...

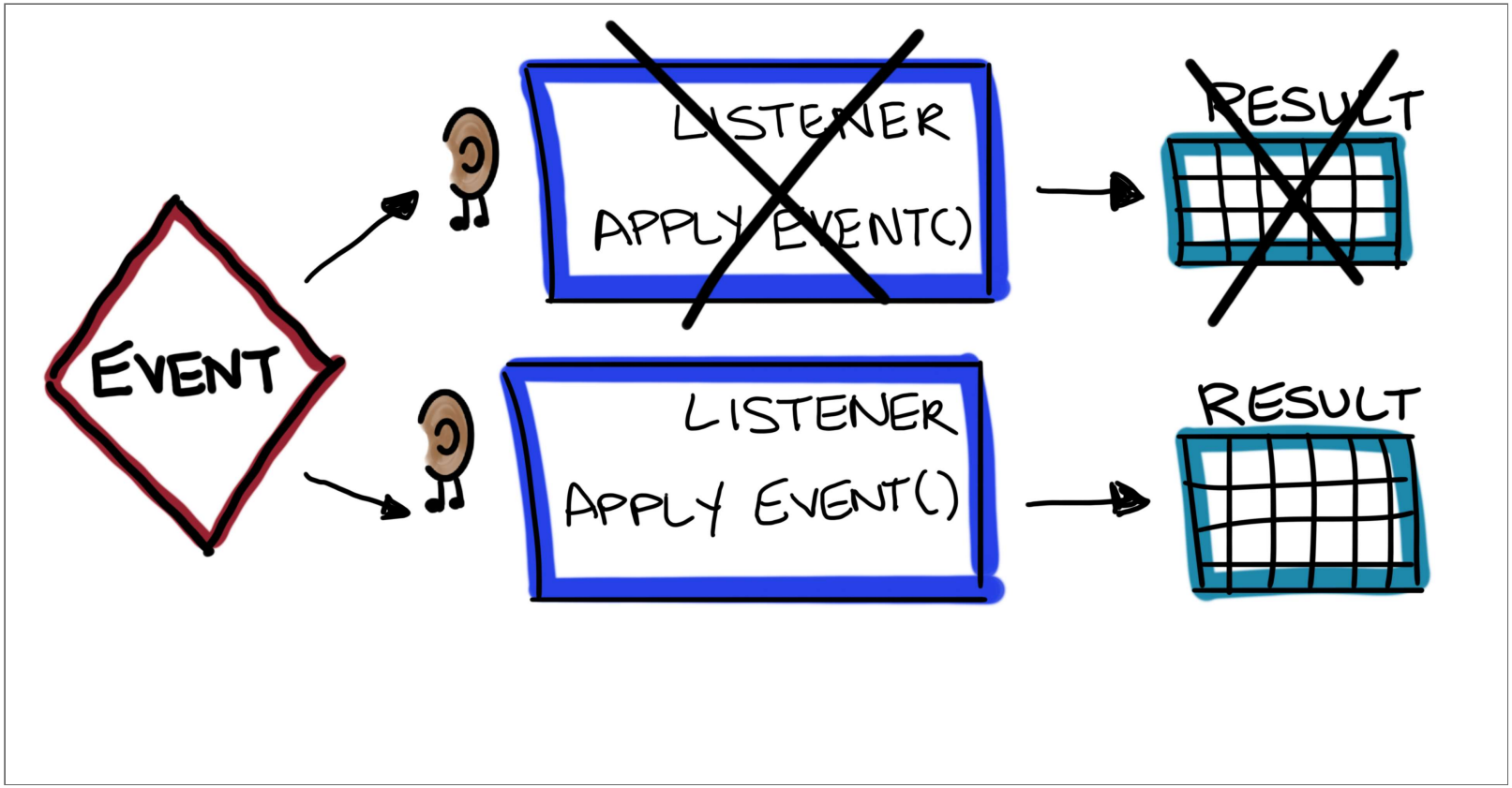


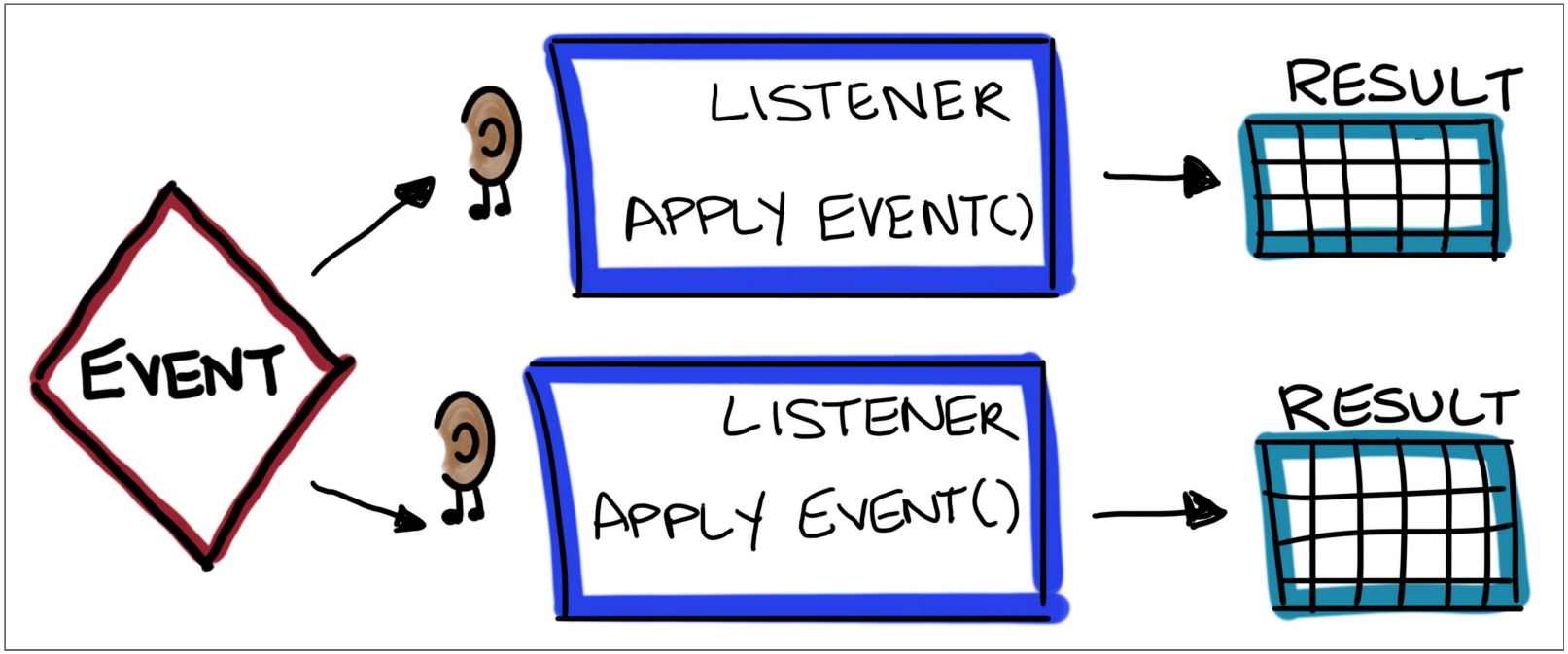

YOU WERE SO PREOCCUPIED WITH WHETHER OR NOT YOU COULD
YOU DIDN'T STOP TO THINK IF YOU SHOULD

EVENTS RARELY CHANGE

- The part of the code that will change is most likely the **result that follows that event**.
- The **structure of the resulting data** is more likely to change than the thing that happened

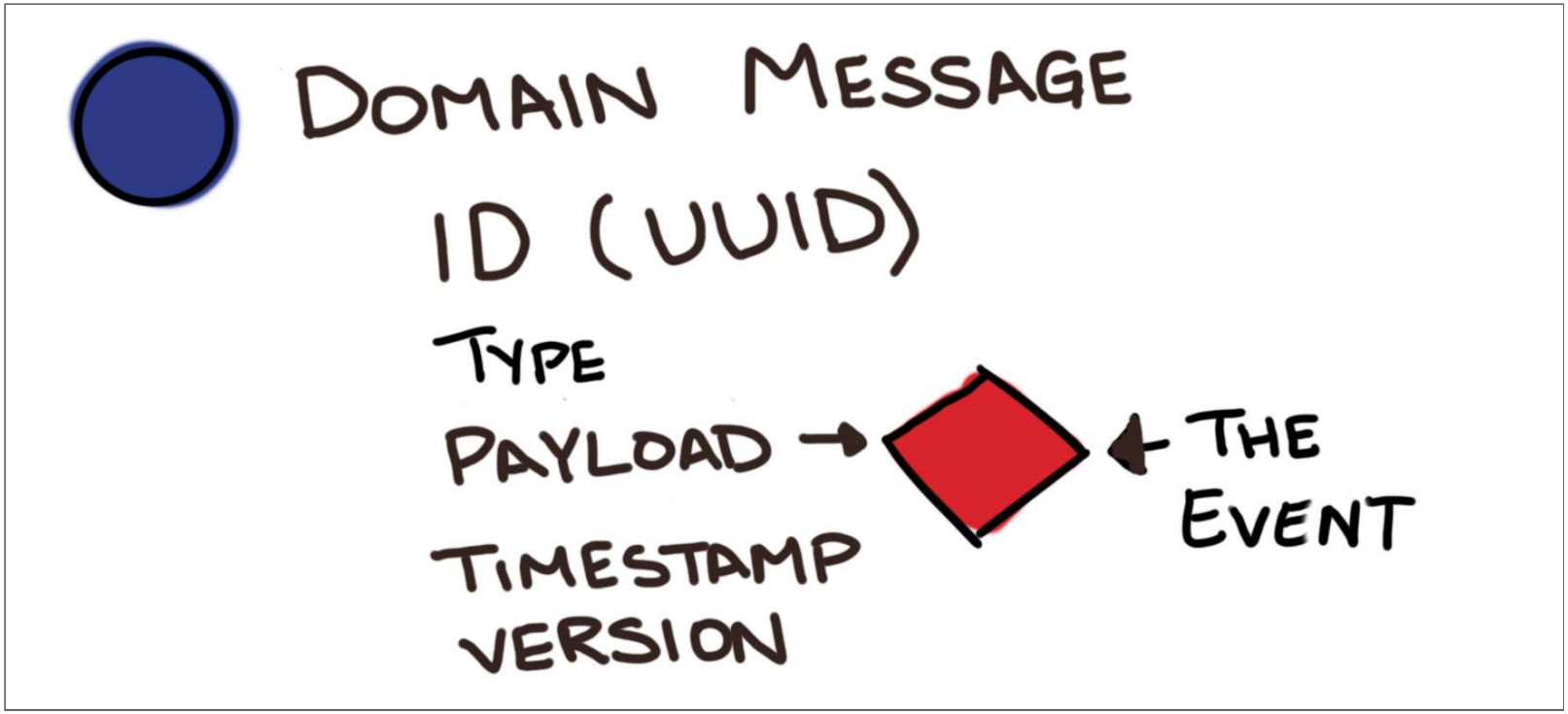

EVENT
LISTENER
APPLY EVENT()
RESULT


EVENT
LISTENER
APPLY EVENT()
RESULT
LISTENER
APPLY EVENT()
RESULT


EVENT
LISTENER
APPLY EVENT()
RESULT
LISTENER
APPLY EVENT()
RESULT

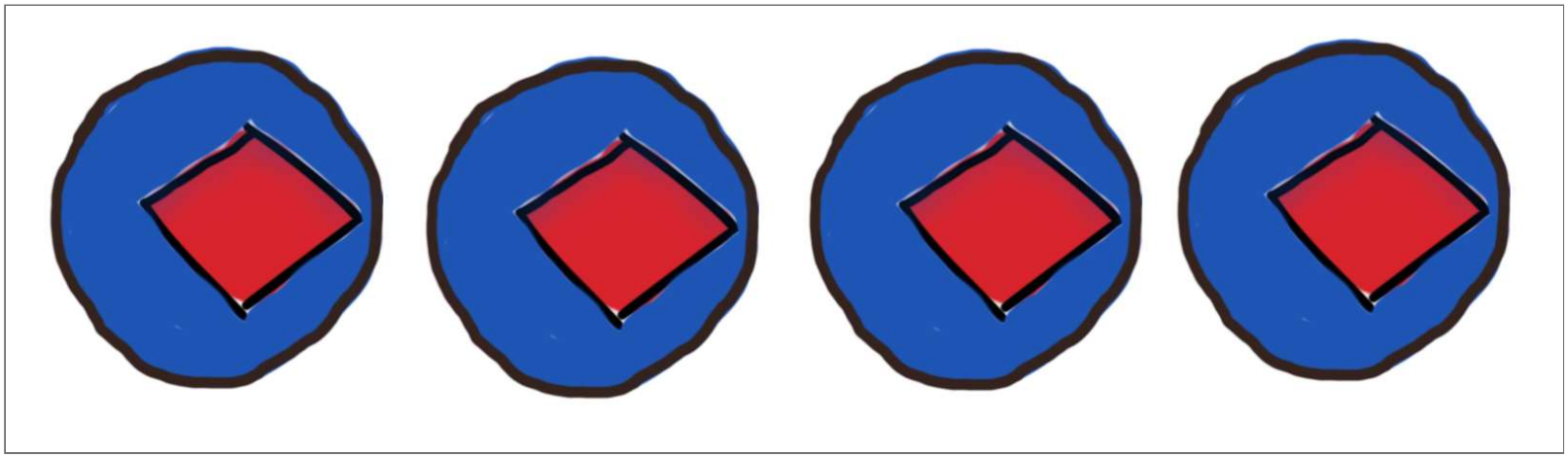
REASONS TO USE EVENTS

- State transitions are important
- We need an audit log, proof of the state we are currently in
- The history of what happened is more important than the current state
- Events are replayable if behavior in your application changes


DOMAIN MESSAGE
ID (UUID)
TYPE
PAYLOAD
TIMESTAMP
VERSION
THE EVENT

EVENT STORE

- Domain-specific database
- Based on a Publish-Subscribe message pattern

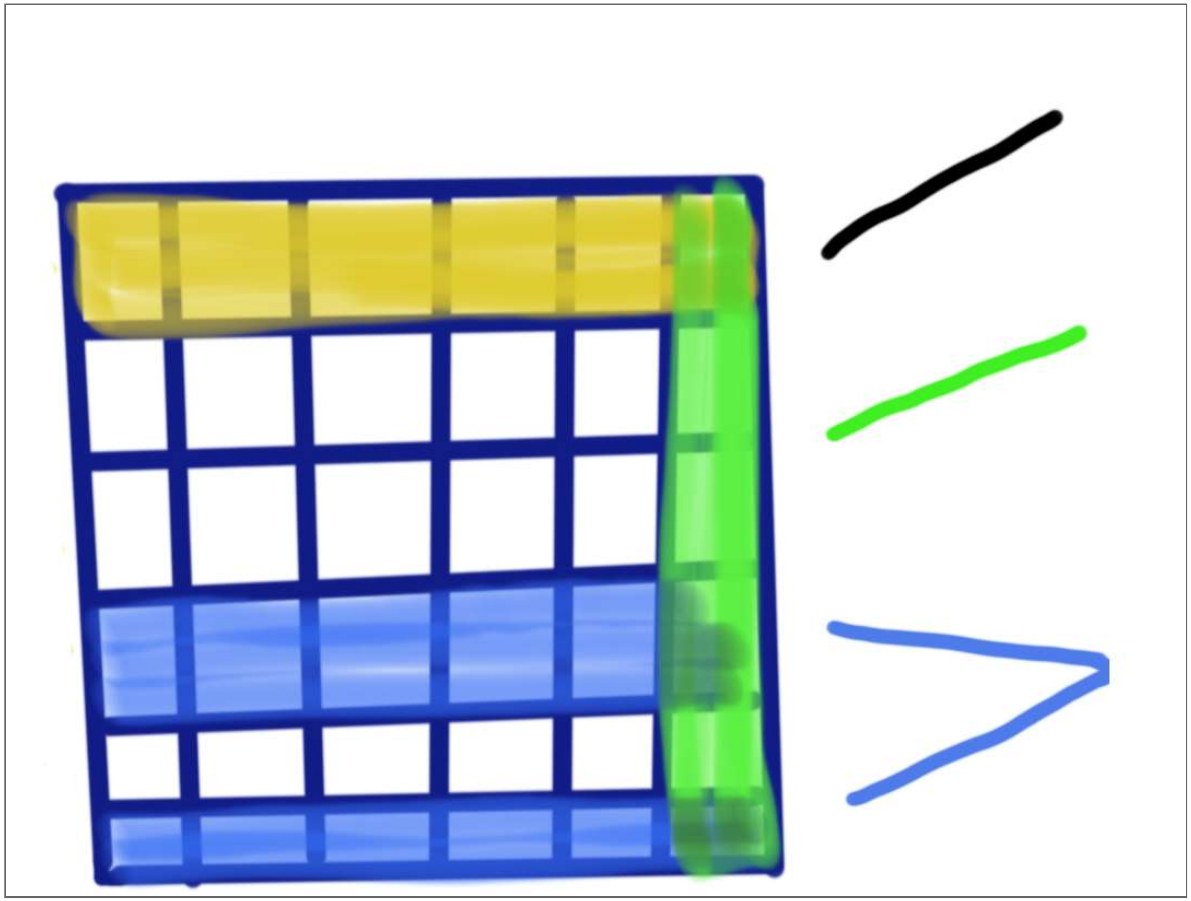


PROJECTOR

```php
<?php

namespace Library\ReadModel;

use Library\Events\BookWasCheckedIn;
use Library\Events\BookAddedToBookshelf;
use App\Support\ReadModel\Replayable;
use App\Support\ReadModel\SimpleProjector;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Connection;

class BookshelfProjector extends SimpleProjector implements Replayable
{

    /**
     * @var Connection
     */
    private $connection;

    /**
     * @var string table we're playing events into
     */
    private $table = 'proj_bookshelf';

    public function __construct(Connection $connection)
```

A set of event handlers that work together to build and maintain a table to be accessed by the read model.

READ MODEL



READ MODEL

```php
<?php

namespace Library\ReadModel;


use Carbon\Carbon;
use Illuminate\Database\Eloquent\Model;

/**
* @codeCoverageIgnore
*/
class Bookshelf extends Model
{
    protected  $table = 'proj_bookshelf';
    public $incrementing = false;
    public $timestamps = false;

    public static function lookupLoansFor($patronId)
    {
        return static::where('patron_id', $patronId)->get();
    }

    public function lookupAvailableBooks()
```

CONNECTING THE EVENTS

- **An event is created only after validation**
  - directly in a controller 'checkout' method
  - using a Check Out Book Command and Handler

CQRS

Command and Query Response Segregation

- **Command** is any method that mutates state
- **Query** is any method that returns a value
- These methods become part of Services
- When parts of your application no longer fit a CRUD model
- Should only be used on specific portions of a system, not the system as a whole

COMMAND HANDLER

A command handler receives a command and brokers a result from the appropriate aggregate. "A result" is either a successful application of the command, or an exception.

should affect one and only one aggregate

COMMAND HANDLER

1. Validate the command on its own merits.
2. Validate the command on the current state of the aggregate.
3. If validation is successful, create an event(s)
4. Attempt to persist the new events. If there's a concurrency conflict during this step, retry or exit.

OPTIMIZE

- to optimize your application for reads and writes
  - all **commands** go into a WriteService
  - all **queries** go into a ReadService
- Separate the load from reads and writes allowing you to scale each independently.

YOU CAN CHOOSE WHICH IS BEST

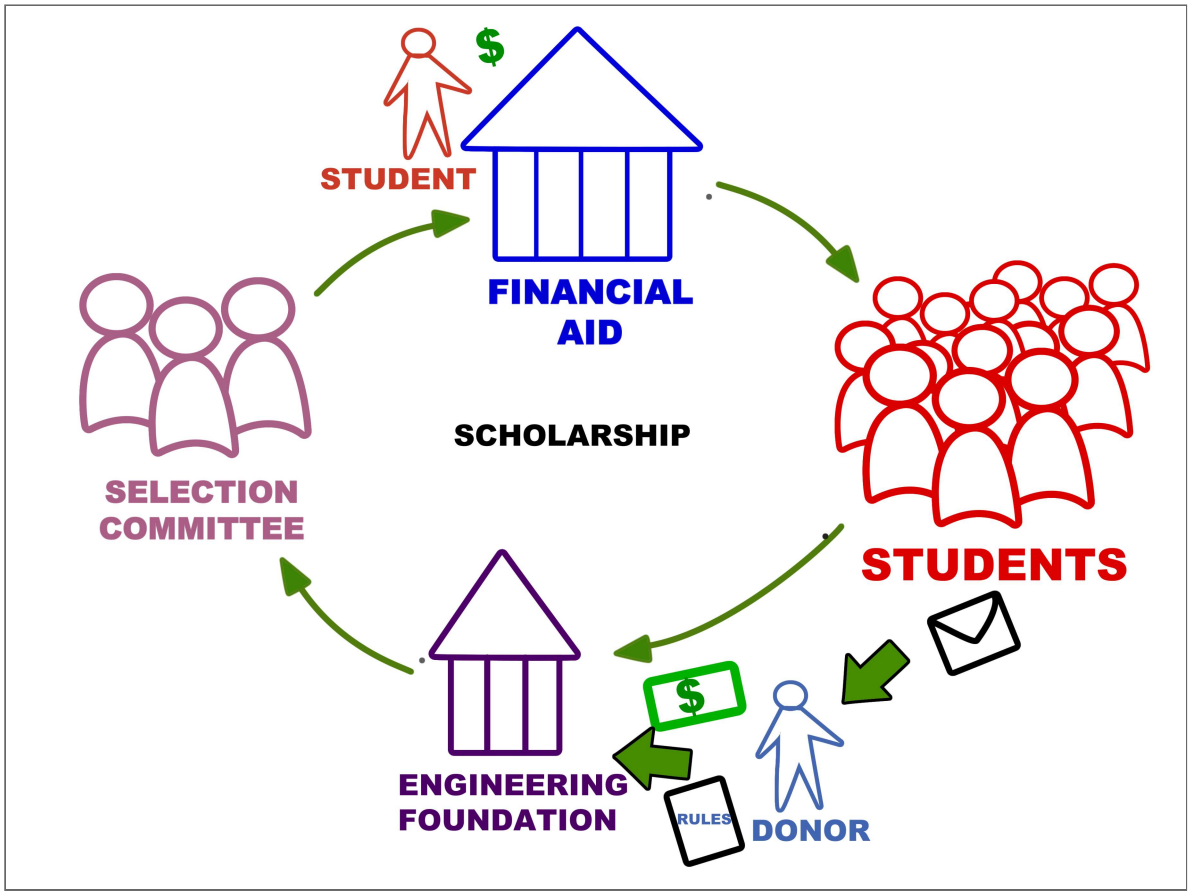
- CRUD
- Event-Sourcing
- Event-Sourcing with CQRS
- Event-Sourcing, CQRS, DDD

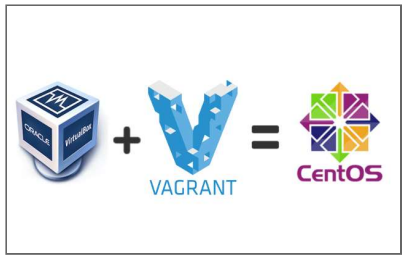
MY PROJECTS

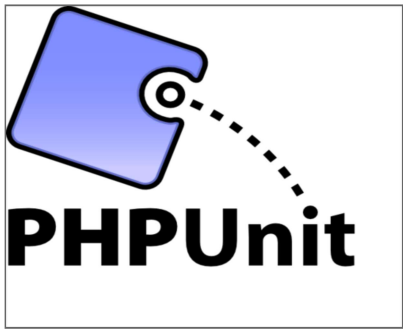
- Scholarships
- Course Registration
- Threat Reports

SCHOLARSHIPS

- replaced legacy Code Igniter in pieces
- added DDD & ES & CQRS beside complex framework spaghetti
- created separate domains
- cleaner history and reports of what had been done

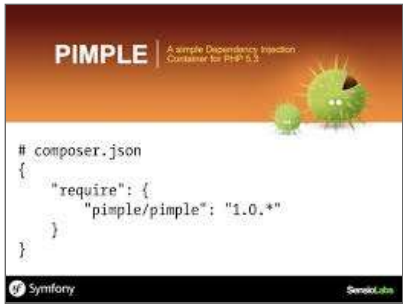

$
STUDENT
FINANCIAL AID
SCHOLARSHIP
SELECTION COMMITTEE
STUDENTS
$
ENGINEERING FOUNDATION
RULES
DONOR


VAGRANT
CentOS

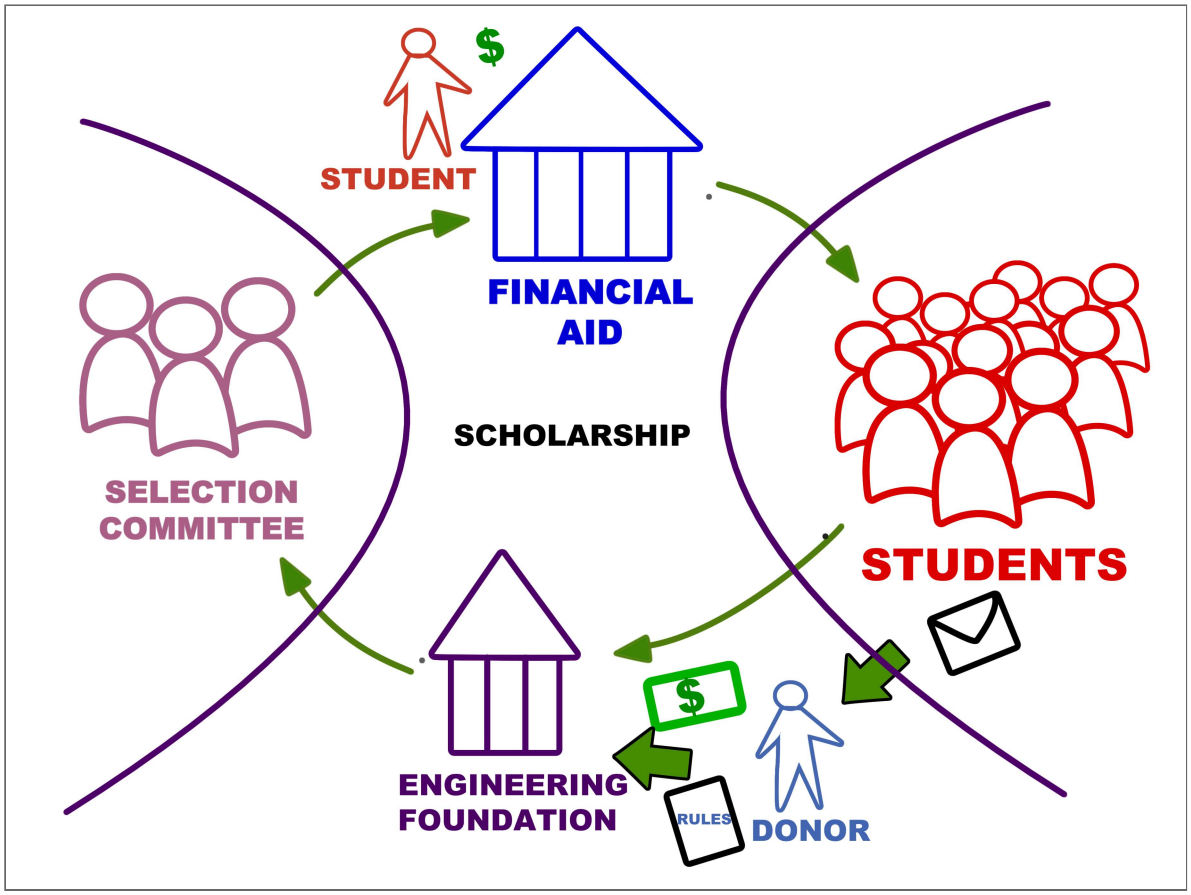

PHPUnit

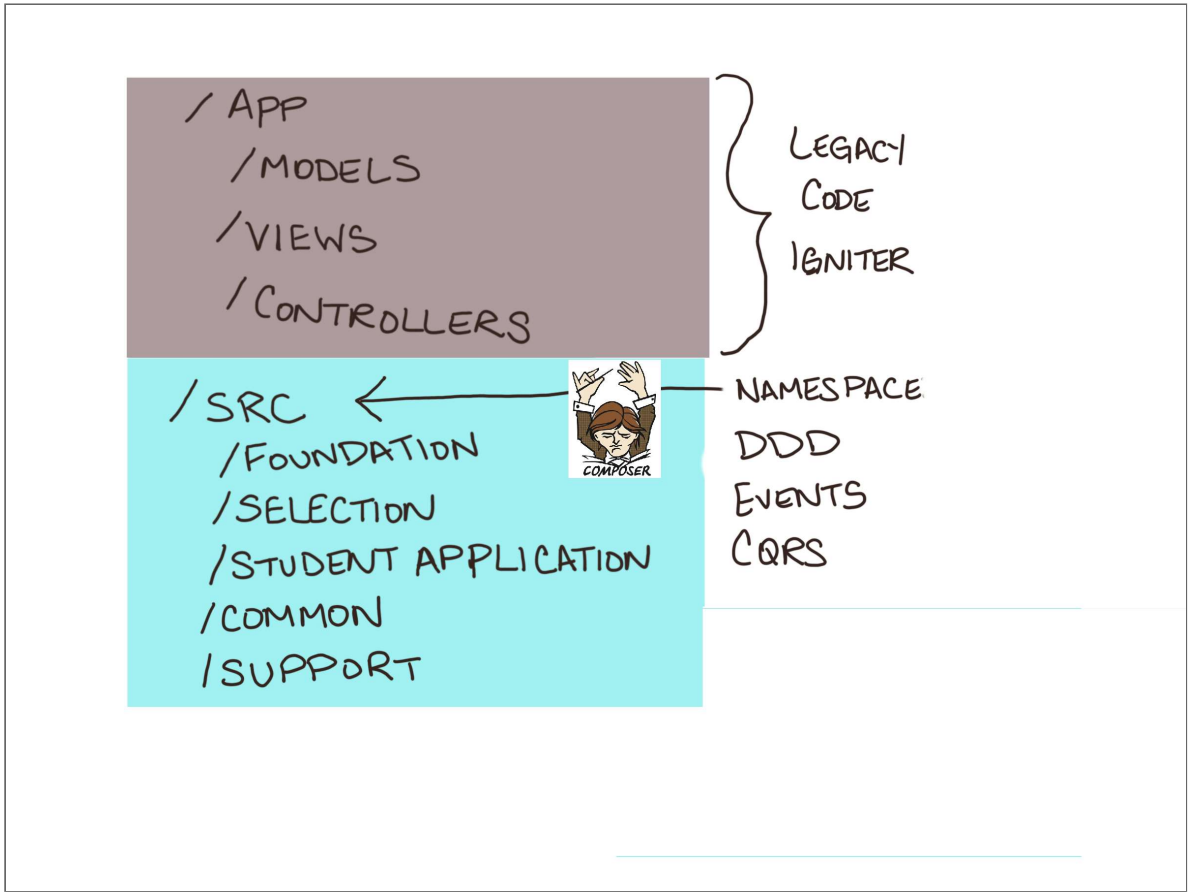

COMPOSER


TWIG


PIMPLE
A simple Dependency Injection Container for PHP 5.3
# composer.json
{
"require": {
"pimple/pimple": "1.0.*"
}
}
Symfony
SensioLabs


/APP
/MODELS
/VIEWS
/CONTROLLERS
LEGACY
CODE
IGNITER

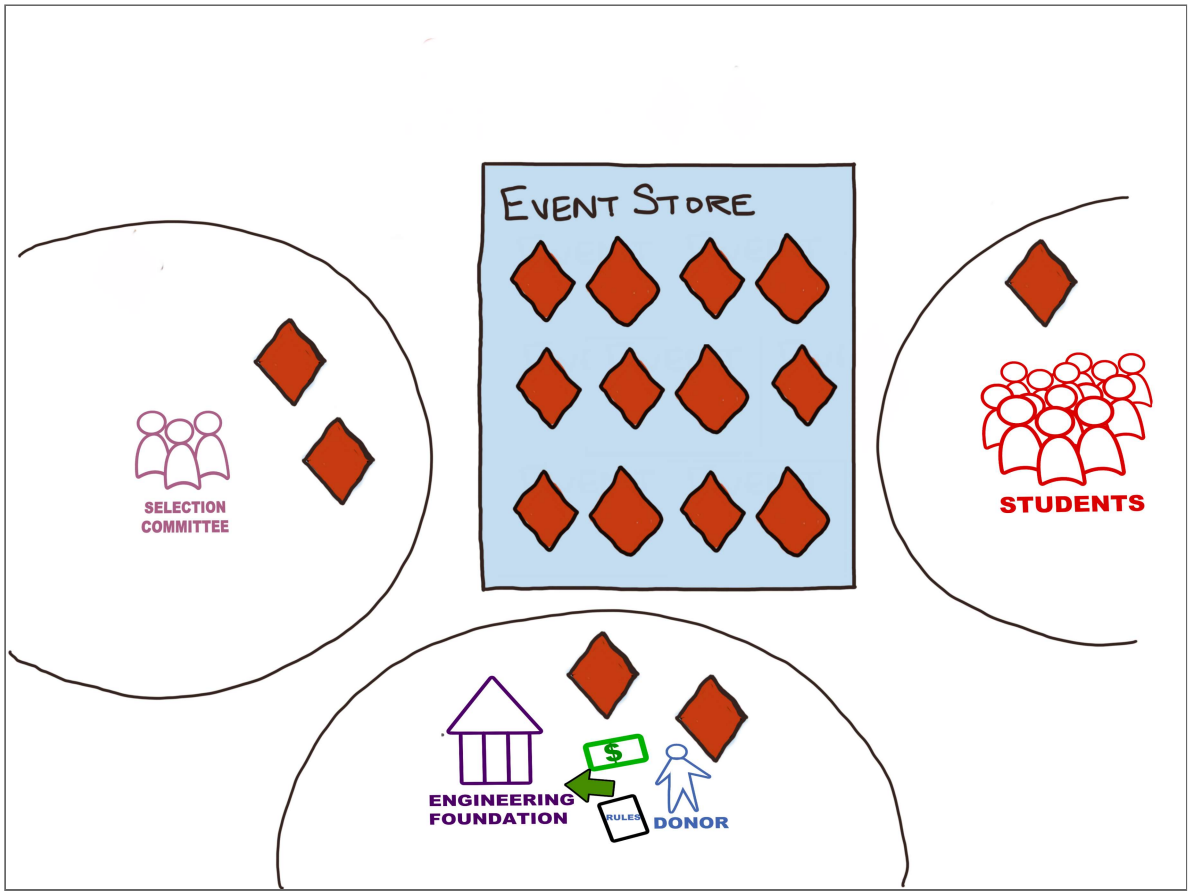

$
STUDENT
FINANCIAL AID
SCHOLARSHIP
SELECTION COMMITTEE
STUDENTS
$
ENGINEERING FOUNDATION
RULES
DONOR

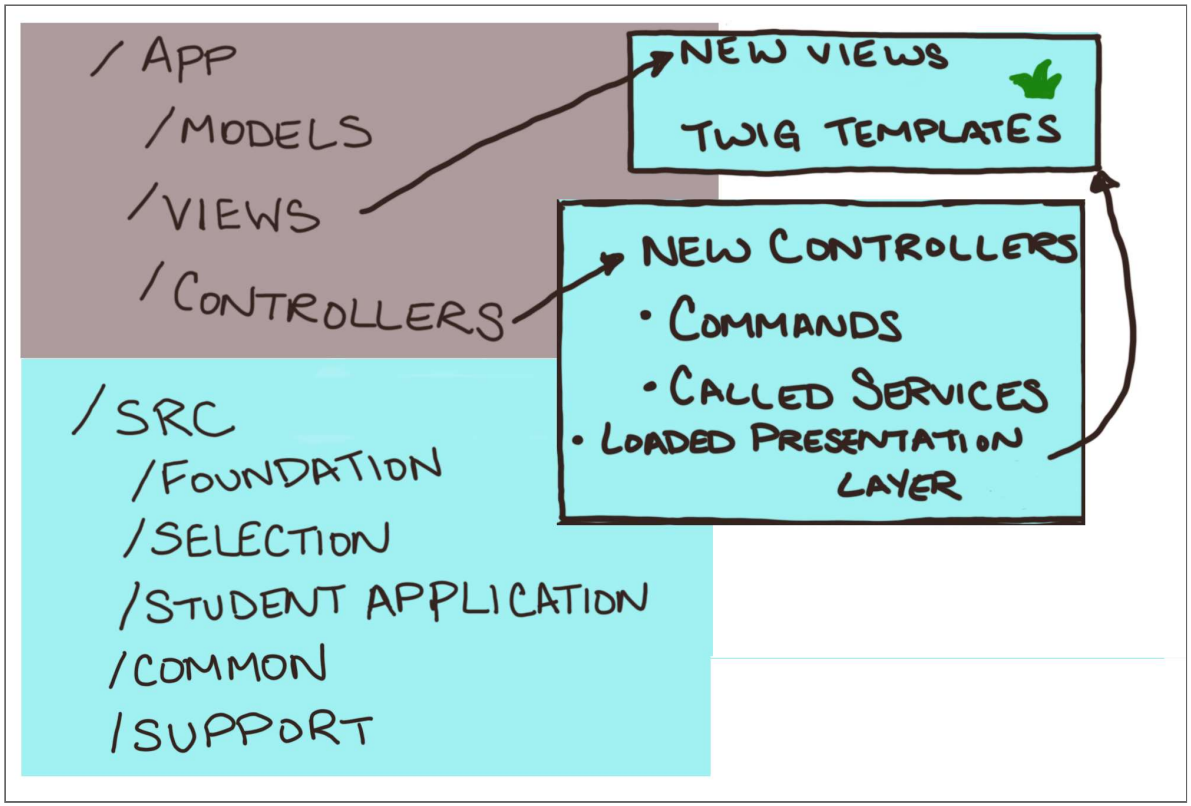

/APP
/MODELS
/VIEWS
/CONTROLLERS
LEGACY CODE
IGNITER
/SRC
/FOUNDATION
/SELECTION
/STUDENT APPLICATION
/COMMON
/SUPPORT
COMPOSER
NAMESPACE
DDD
EVENTS
CQRS

BOOTSTRAPPING: NAMESPACES

- '/src' is given a namespace
- namespaces are autoloaded in composer

```
{
    "name": "itecs/scholarships",
    "description": "Scholarships application for the College of Engineering.",
    ...
    "autoload": {
        "psr-4": {
            "ITECS\\Scholarships\\": [ "src/", "app/core/" ],
            "Codeception\\Module\\": "src/",
            "Tests\\Substitute\\": "tests/_helpers/"
        }
    }
}
```


EVENT STORE
SELECTION COMMITTEE
STUDENTS
ENGINEERING FOUNDATION
RULES
DONOR


/ APP
/ MODELS
/ VIEWS
/ CONTROLLERS
NEW VIEWS
TWIG TEMPLATES
NEW CONTROLLERS
• COMMANDS
• CALLED SERVICES
• LOADED PRESENTATION LAYER
/ SRC
/ FOUNDATION
/ SELECTION
/ STUDENT APPLICATION
/ COMMON
/ SUPPORT

DEPENDENCY INJECTION

DEPENDENCIES

The dependencies are the objects your class needs to function

GPAService needed a DB connection

INJECT THE DEPENDENCIES

```
function __construct(Database $database)
{
    $this->database = $database;
}
```

DEPENDENCY INJECTION

- Decouples our code from low level implementation details
- Instantiate our classes with their dependencies
- AND instantiate those dependencies

DEPENDENCY MAP

'/app/bindings.php'

```
<?php

    /* example one */

    use Scholarships\Selection\Services\IlluminateDatabaseGpaService;

    $container['Scholarships\Common\Services\GpaService'] = function($c) {
        return new IlluminateDatabaseGpaService($c['database']);
    };
```

DEPENDENCY MAP

'/app/bindings.php'

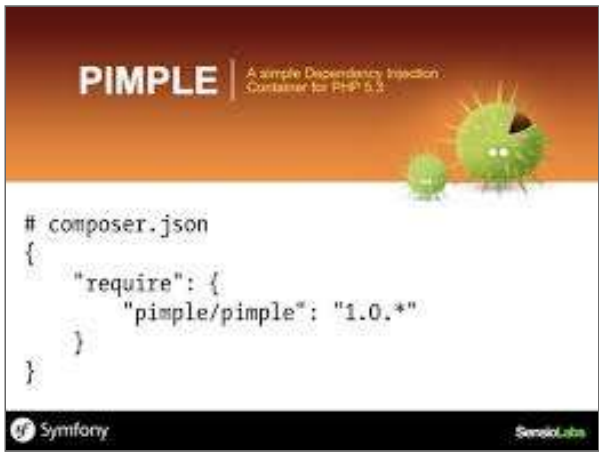
```
<?php

/* example two */

use Scholarships\Selection\ApplicantQueryService;

$container['Scholarships\Selection\ApplicantQueryService'] = function($c) {
    return new ApplicantQueryService(
        $c['Scholarships\StudentApplication\StudentQueryService'],
        $c['Scholarships\Common\Services\ResidenciesService'],
        $c['Scholarships\Common\Services\GpaService'],
        $c['Scholarships\Common\Services\UnmetNeedService'],
        $c['Scholarships\Common\Services\CandidateQualificationService']
    );
};
```

DEPENDENCY INJECTION CONTAINER

- A map of dependencies your class needs along with the logic to create those dependencies if they haven't been created yet
- Can resolve complex dependencies transparently
- Modular when you need to swap a dependency, only update the container

PIMPLE

- Create your container by instantiating the Pimple class



'APP/SERVICES.PHP'

```
<?php

    use Pimple\Container;
    use Illuminate\Database\Capsule\Manager as Capsule;

    $container = new Container();

    $container['config'] = require_once('config.php');

    $container['database'] = function ($c) {
    $capsule = new Capsule;

    // PHP 5.3 doesn't do array de-referencing.
    // @todo update for php54
    $config = $c['config'];

    $capsule->addConnection(
        array(
            'driver'    => 'mysql',
            'host'      => $config['db']['default']['hostname'],
            'port'      => $config['db']['default']['port'],
            'database'  => $config['db']['default']['database'],
```

CONFIGURATION FILE

- DB connections
- Base URL
- Set paths to twig templates
- Customize Notice messages
- Set config variables for services

CONFIGURATION FILE

'/app/config.php'

```
<?php
    return array(
        /* base url for path in the site */
        'app' => array(
            'base_url' => sprintf('http://localhost:%s/', isset($_SERVER['SERVER_PORT'])
            ? $_SERVER['SERVER_PORT'] : ''),
            'index_page' => 'index.php/',
            'debug' => FALSE
        ),

        /* DB configuration */
        'db' => array(
            'default' => array(
                'hostname' => "local__server",
                'port' => "3306",
                'username' => "uname",
                'password' => "pword",
                'database' => "db_name_"
            )
        ),

        'twig' => array(
```

We passed this array into a **$container['config']** variable

BASE CONTROLLER

```
<?php

namespace ITECS\Scholarships;

use \ReflectionClass;
use \Log;

/**
* Base Application Controller Class
*
*/
class BaseController extends \Controller {

    const BAD_REQUEST=400;
    const FORBIDDEN=403;
    const NOT_FOUND=404;
    const METHOD_NOT_ALLOWED=405;
    const SERVER_ERROR=500;

    protected $serverResponseCodes;
    protected $authenticationService;
```

CONNECTING TO THE FRAMEWORK

- '/app/bindings.php'
  the roadmap of our new code and dependencies
  required in the '/app/services.php' file
- '/app/services.php'
  defined and configured twig, database, et al
  required in the CI index.php file
- '/app/controllers'
  new controllers for the new functionality

CONTROLLERS

```
<?php

class Selectionnext extends BaseController
{
    public function Selectionnext()
    {
        parent::BaseController();

        $this->scholarshipRepository = $this->container['Scholarships\Selection\Scholar
        $this->collaborationsService = $this->container['Scholarships\Selection\Collabo
        $this->committeeService = $this->container['Scholarships\Selection\CommitteeSer
        $this->authService = $this->container['Scholarships\IdentityAccess\Authenticati
        $this->awardsService = $this->container['Scholarships\Selection\Scholarship\Awa
        $this->events = $this->container['Scholarships\Support\Events\EventStore'];
    }

    private function getScholarshipDashboard($scholarshipId,$keepItLight=false)
    {
        $committeeMember = $this->getCommitteeMember();

        $presenter = new ScholarshipDashboardPresenter(
            $committeeMember,
```

SCHOLARSHIPS WRAP-UP

- this was an example of replacing the code in pieces
- could have replaced only one part or a few
- really powerful to see which parts of your application could benefit from event sourcing
- and finding the simplest way to implement

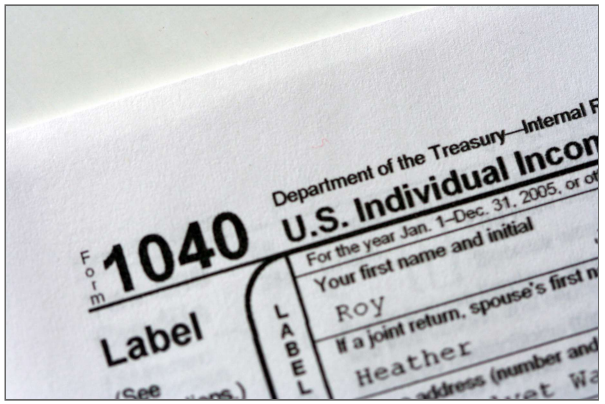
STUDENT ENROLLMENT PROCESS

- rewrote
- ES to follow the process
- status drop-down versus events

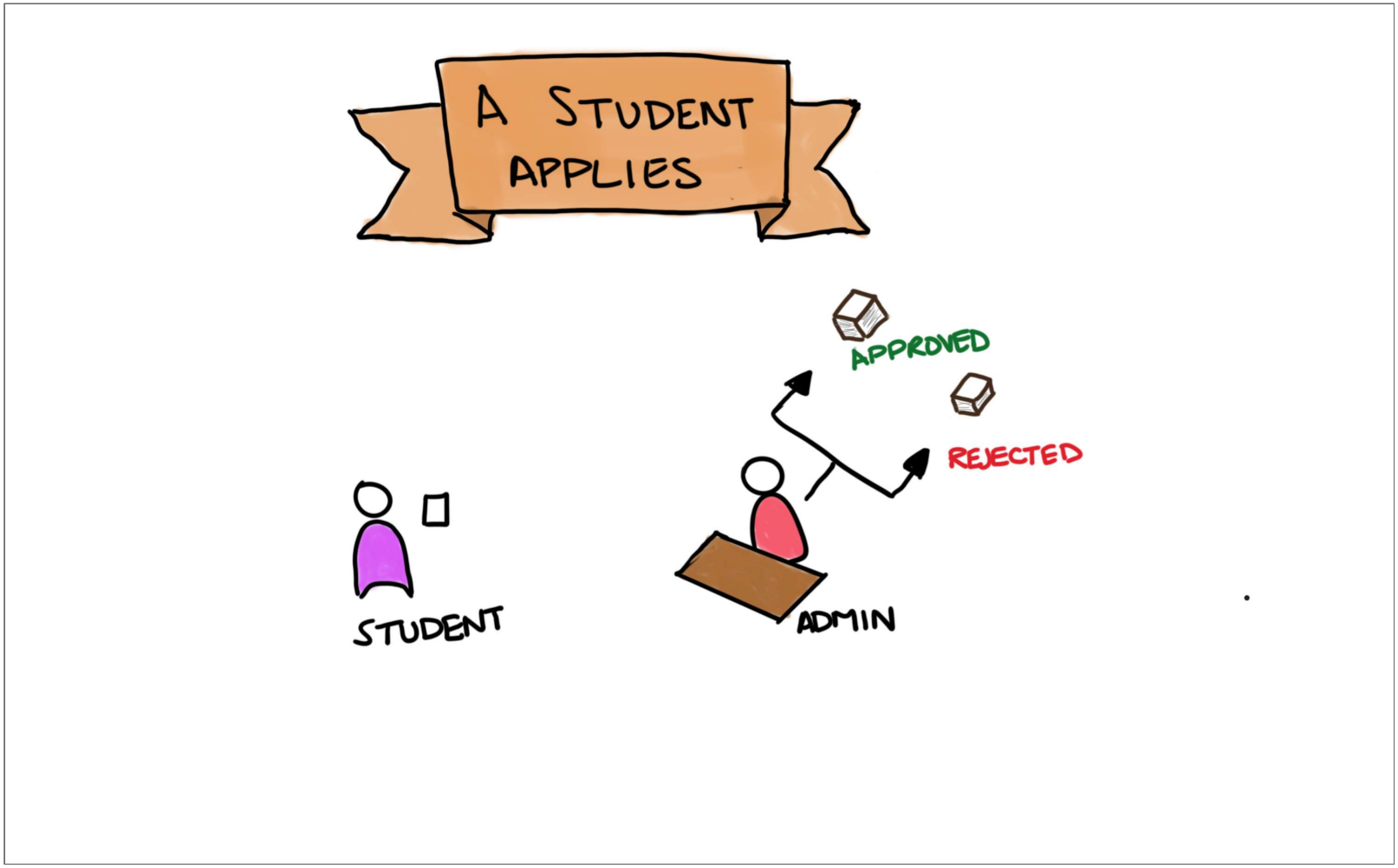
BEGAN WITH PAPER FORMS

A lot of our systems were built to replace paper processes

They often closely map to this physical form.



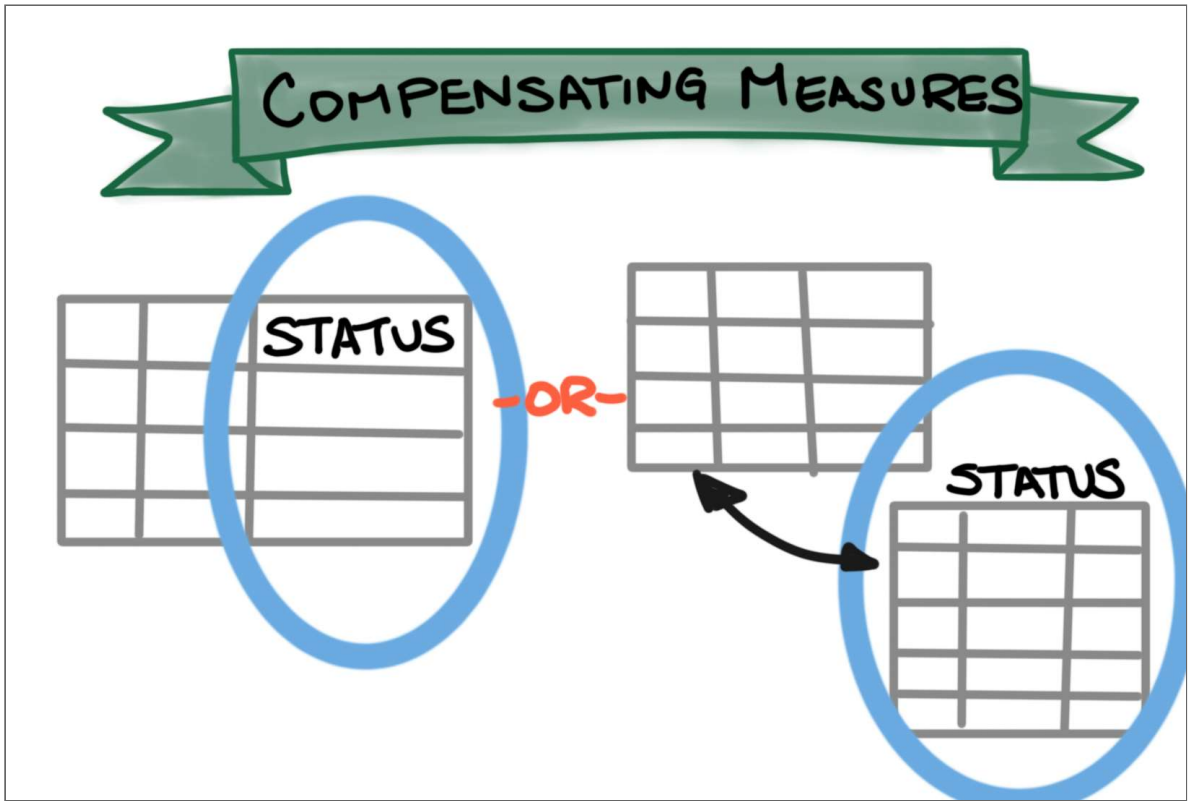
© PHOTO BY AIDAN MORGAN


A STUDENT APPLIES
APPROVED
REJECTED
STUDENT
ADMIN

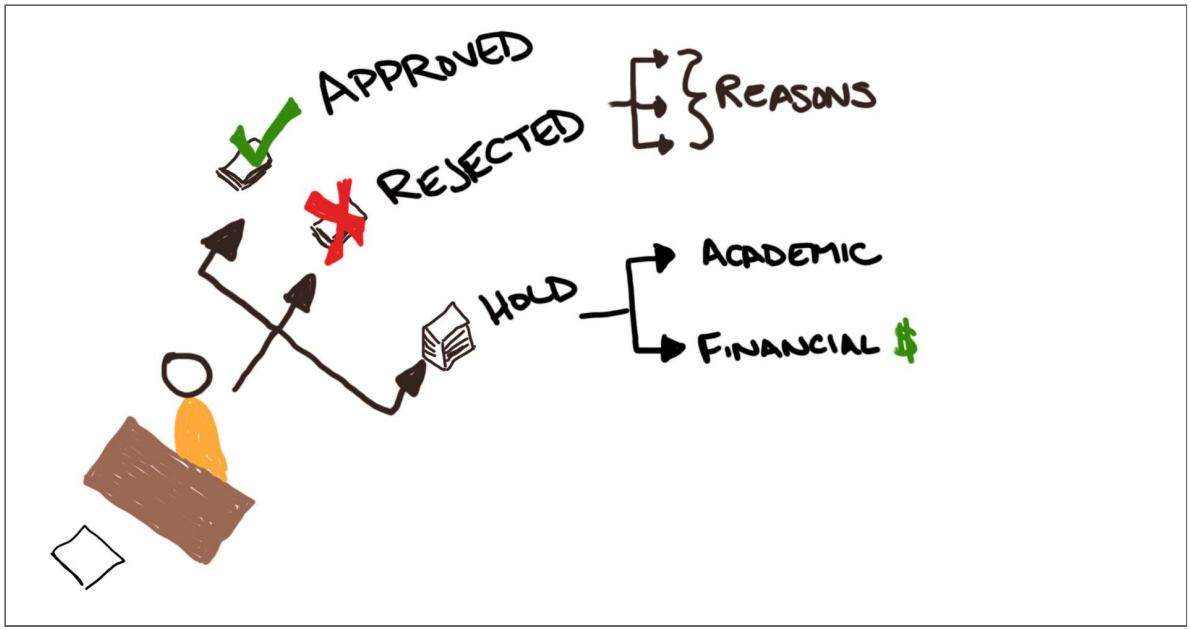
PAPER FORMS HANDLING STATE

Status labels are like a rubber stamp

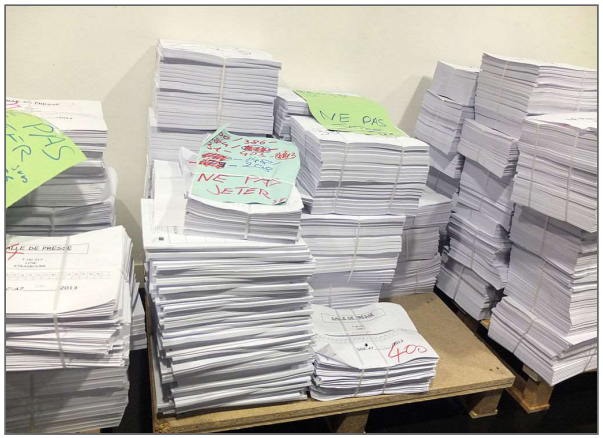
Status doesn't always communicate why or what happened


COMPENSATING MEASURES
STATUS
-OR-
STATUS

HOW WORKFLOWS BECOME COMPLEX



PAPER FORMS HANDLING STATE



Piles indicate status of the form

A SIMPLE STATUS DROP-DOWN

RECEIVED ▼

A "SIMPLE" STATUS DROP-DOWN

RECEIVED
CANCELLED
COURSE CANCELLED
COURSE NOT APPROVED
CPC DENIED
CPC NOT APPROVED
CPC PENDING TRANSCRIPT
CPC PROCESSING
DENIED
DROPPED AFTER CENSUS
DROPPED BEFORE CLASSES BEGUN
DROPPED BETWEEN BEGINNING OF CLASS AND CENSUS DATE
DROPS/WITHDRAWALS AT SITES
ECE ON CAMPUS STUDENTS
ENROLLMENT CANCELLED
EOL APPROVED
EOL MISC
NEW STUDENT REGISTRATION
PENDING ON CAMPUS REQUEST
ON CAMPUS STUDENT NOT APPROVED
PENDING CASHIER HOLD
PENDING EOL APPROVAL
PENDING INSTRUCTOR APPROVAL
PENDING INTERNATIONAL STUDENT
PENDING NDS OPEN ENROLLMENT
PENDING OIS VISA STUDENT
PENDING PERMANENT RESIDENT
PENDING TERM ADVISEMENT HOLD
PENDING TRANSCRIPT
PENDING TUITION PREPAYMENT
PREAPPROVED RETURNING
PROCESSING NDS
PROCESSING SITE
PROCESSING Z
PROJECT MESSAGE - MAE 586
PROJECT MESSAGE - NE 693
REGISTERED

REGISTERED
REGISTERED ASHEVILLE
REGISTERED HAVELOCK
REGISTERED WILMINGTON
SITE APPROVAL: ASHEVILLE
SITE APPROVAL: HAVELOCK
SITE APPROVAL: WILMINGTON
SITE NOT APPROVED
SWAPPED OUT
TRANSCRIPT APPROVED
WITHDRAWN

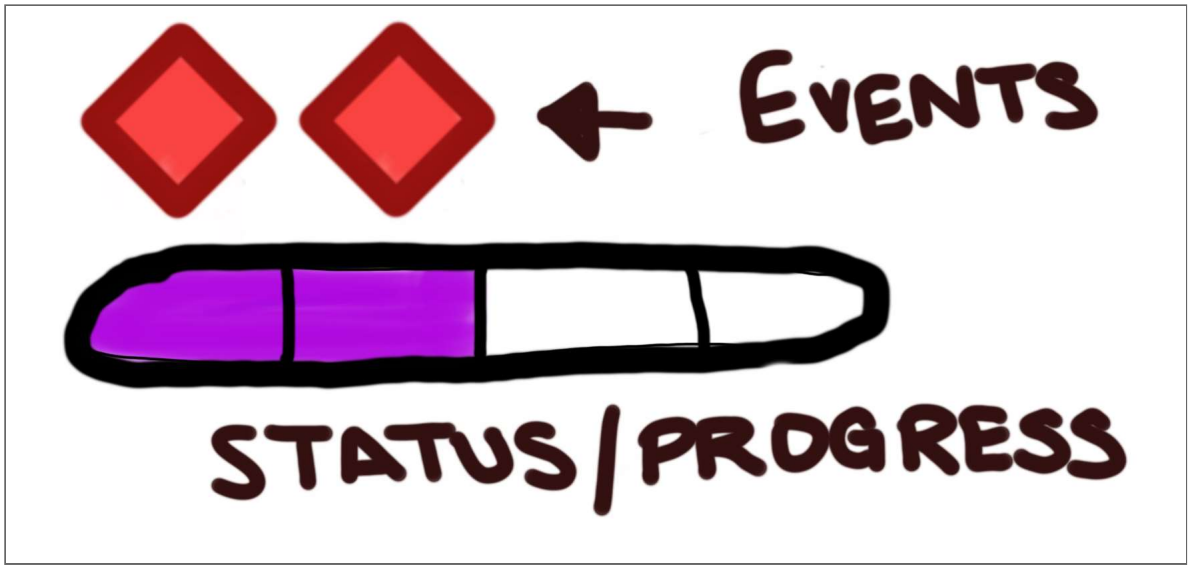
SOMETHING HAPPENED

Status is a reflection of something that happened

There is ONE of each status + reasons/details

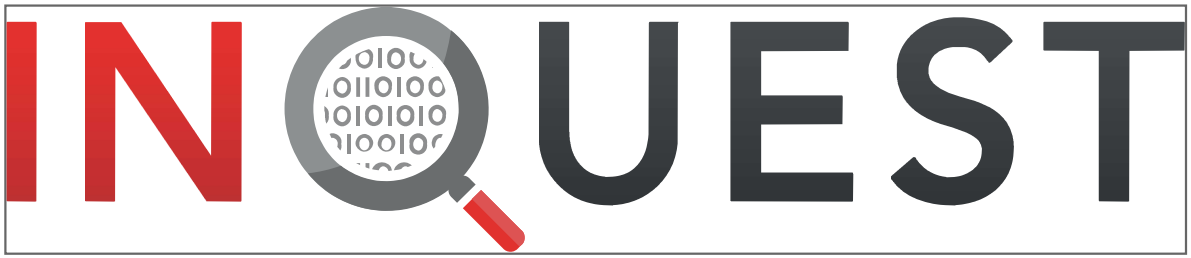
Events can record what happened

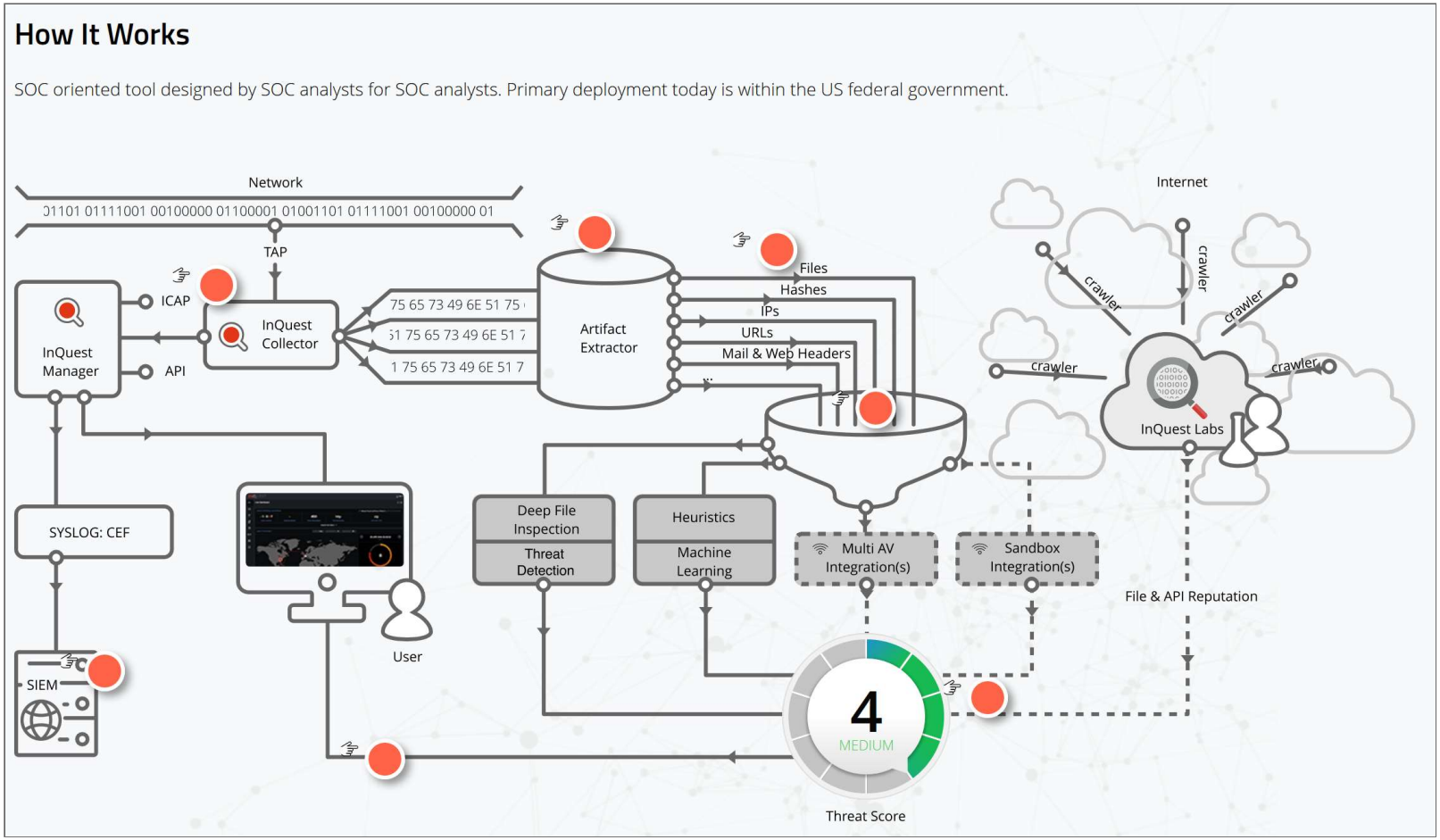
THE REQUEST DOESN'T GET STUCK



STREAMLINED PROCESS

- Focused our customers on their process and provided tools to simplify their jobs
- Managed the process flow and controlled when the student was notified about changes
- Managed the process flow and controlled when the student was notified about changes


INQUEST

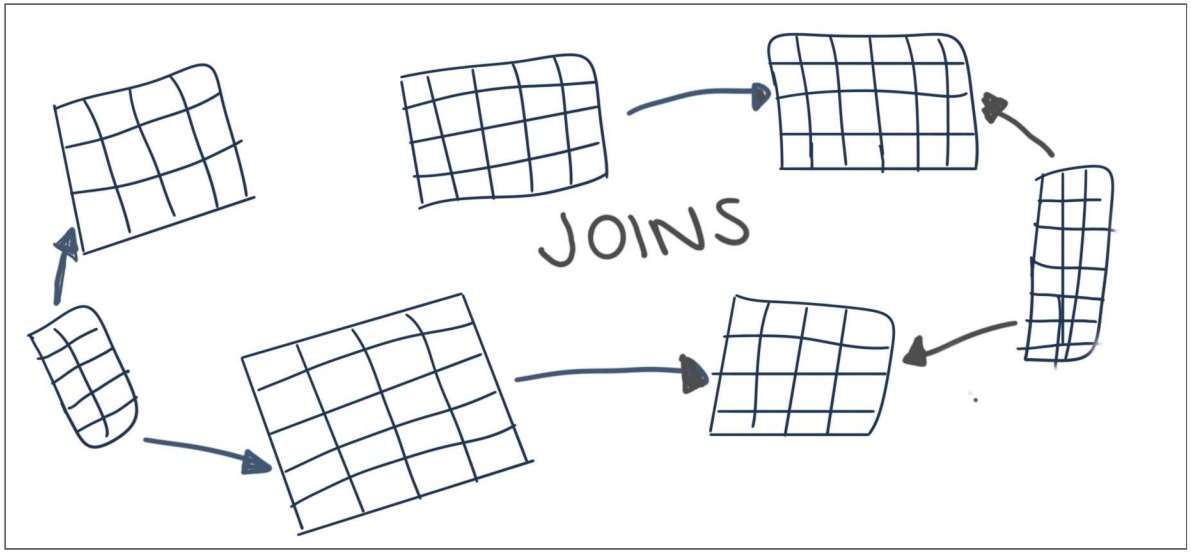

How It Works
SOC oriented tool designed by SOC analysts for SOC analysts. Primary deployment today is within the US federal government.
Network
01101 01111001 00100000 01100001 01001101 01111001 00100000 01
TAP
ICAP
API
InQuest Manager
InQuest Collector
75 65 73 49 6E 51 75
1 75 65 73 49 6E 51 7
Artifact Extractor
Files
Hashes
IPs
URLs
Mail & Web Headers
Internet
crawler
InQuest Labs
SYSLOG: CEF
SIEM
User
Deep File Inspection
Threat Detection
Heuristics
Machine Learning
Multi AV Integration(s)
Sandbox Integration(s)
File & API Reputation
4
MEDIUM
Threat Score

REPORTS

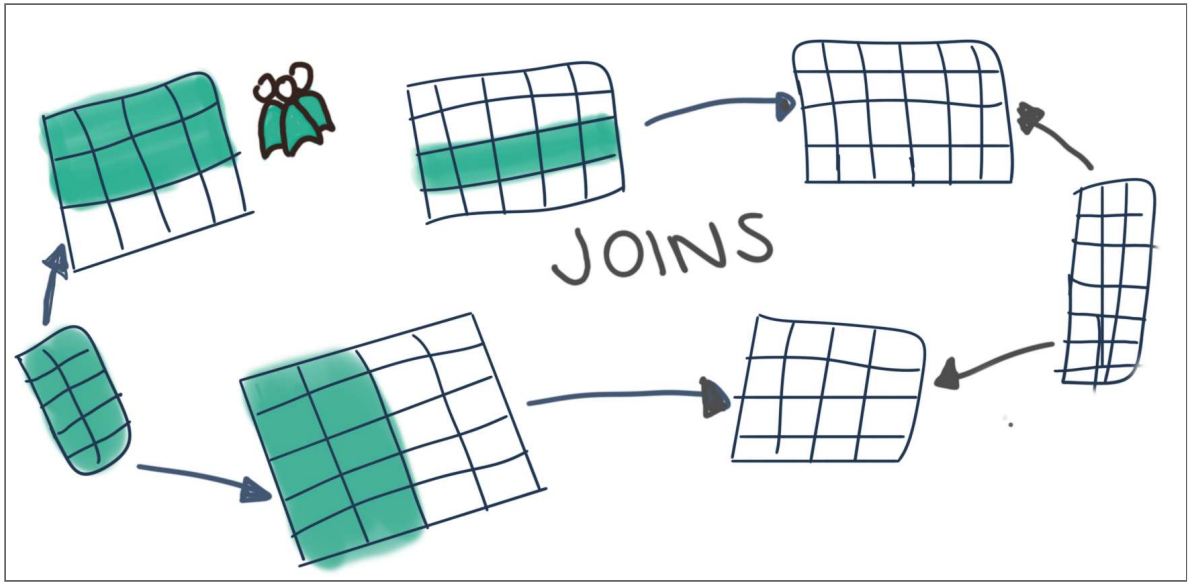
E: Threat: Top Threat Score By File Type (Previous 24 Hours)

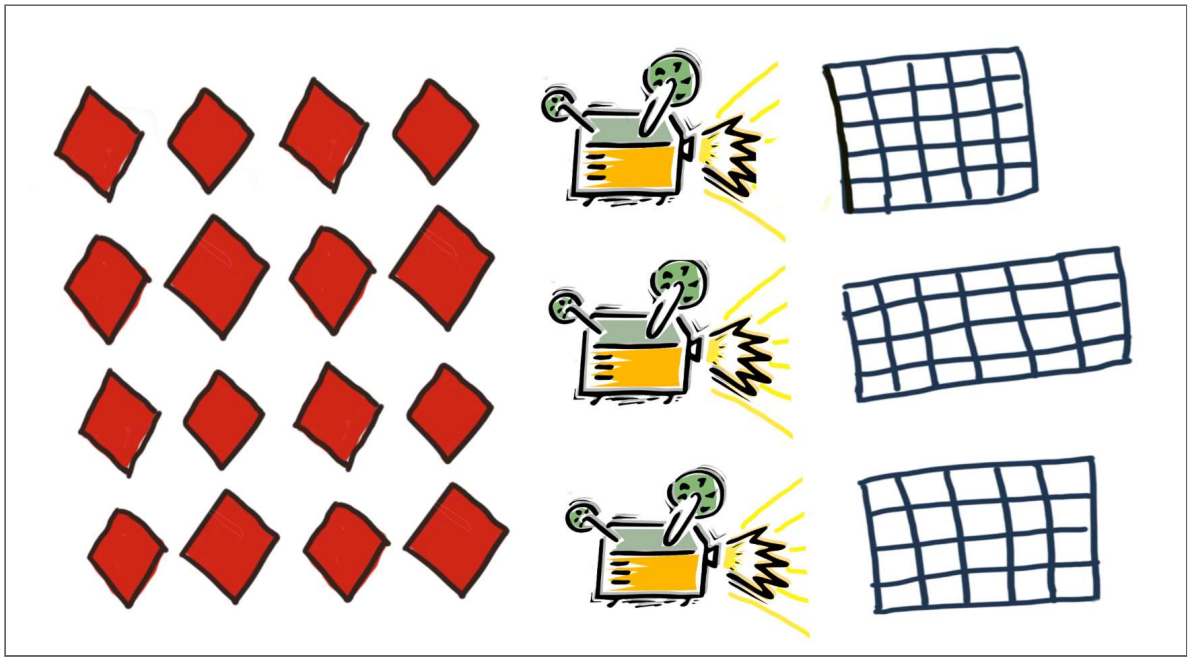
ID	Count	Threat Score	MIME Type
1	129	8	application/x-as400attachment
2	124	9	application/x-unknown
3	99	3	text/x-shellscript
4	82	8	text/rtf
5	79	4	text/html
6	48	7	application/zip
7	42	3	application/x-shockwave-flash
8	29	8	application/x-executable
9	12	6	application/vnd.ms-office
10	11	3	application/vnd.ms-excel
11	5	3	application/pdf
12	3	4	application/octet-stream
13	2	10	application/msword

REPORTS



REPORTS - FILTERED FOR USER ACCESS





PLANNED IMPROVEMENTS

- No more complex DB queries
- Separates us from older schema
- Events can be passed from the Engine to Customer-facing Site
- full audit log of what has happened in our environment
- the results of those events can be optimized for read to shorten the time to retrieve data
- separate the logic of what an event means based on context and purpose
- only display results that a user is allowed to see
- flexible to change, our interpretation of events can change, and we can rebuild projections without losing the full history

МНОГО ВАМ ХВАЛА

EMILY STAMEY

@ELSTAMEY

HTTP://ELSTAMEY.COM

New Stuff to check out!

- **That Podcast Episode 50:** The One Where We Talk to Shawn about Event Sourcery, CQRS, Event Sourcing and GDPR

RESOURCES

- Tools
  - **Event Sauce - Event-sourcing & Commands**
  - **Prooph - CQRS and ES**
  - **Broadway - framework for CQRS and ES**

RESOURCES

- **CQRS by Martin Fowler**
- **CQRS by Greg Young**
- videos
  - **Greg Young CQRS and Event Sourcing**
  - **Greg Young - long class**
  - **Greg Young - A Decade of DDD, CQRS, Event Sourcing**
- podcasts
  - **PHP Round Table - Event Sourcing** (+2 more)