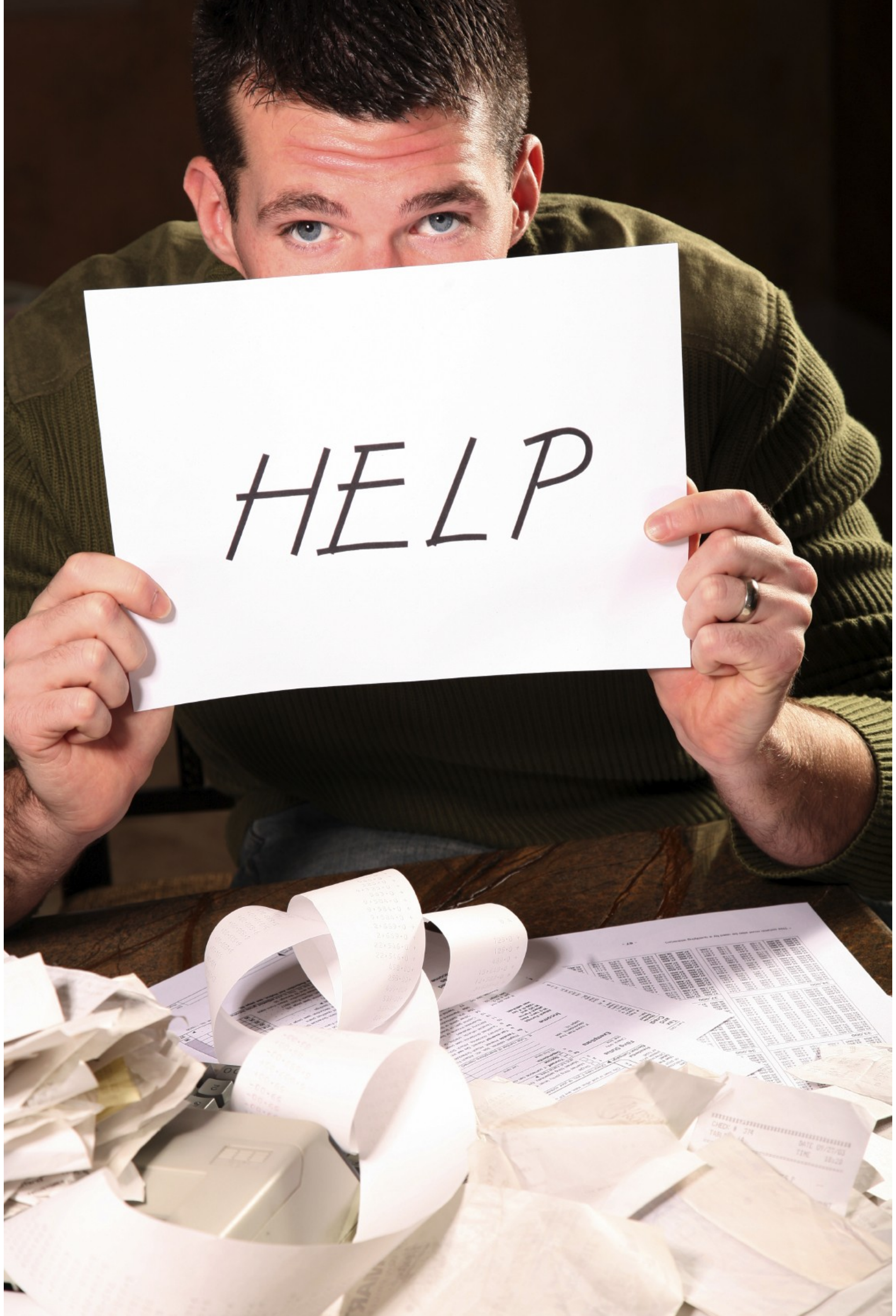
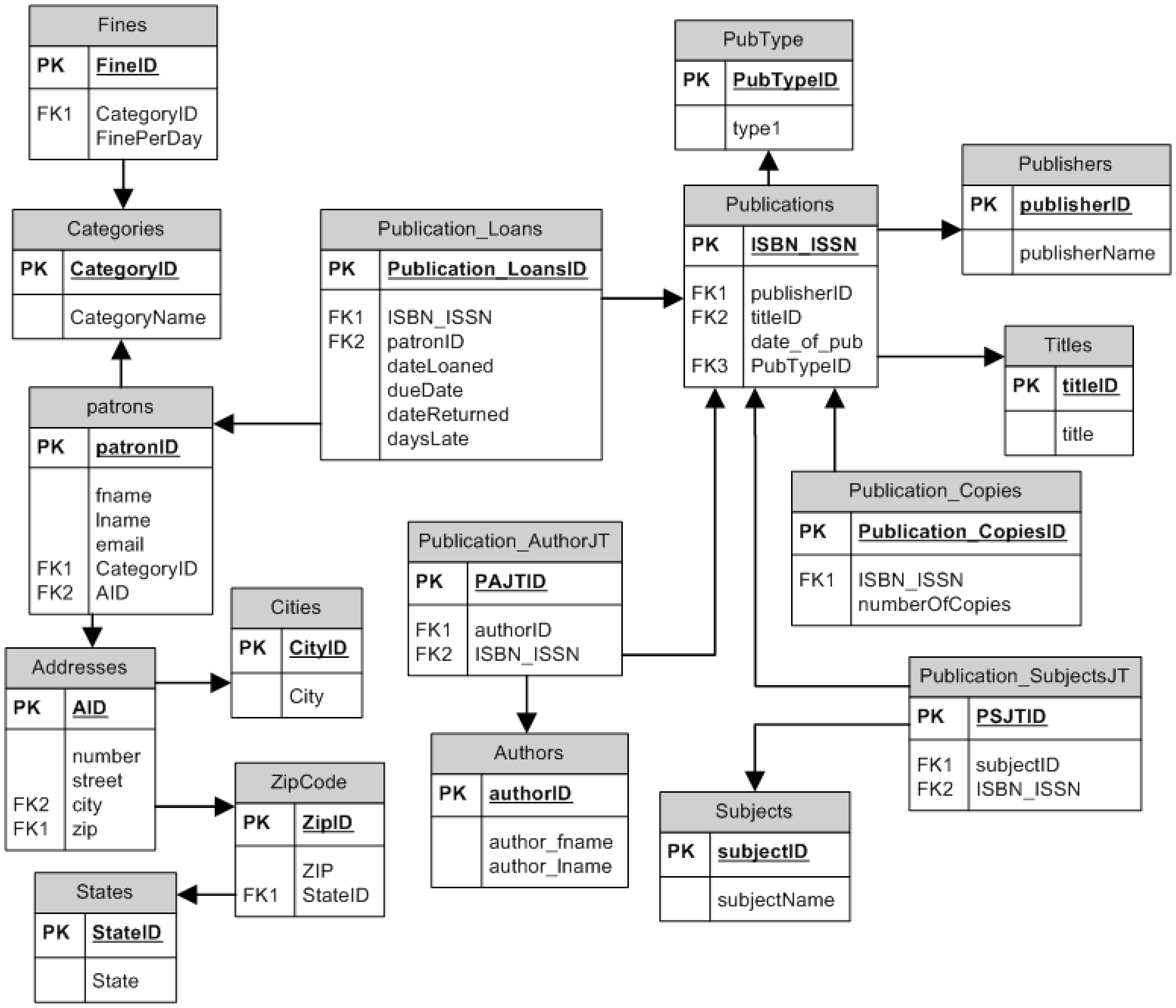
Events are not just for notifications

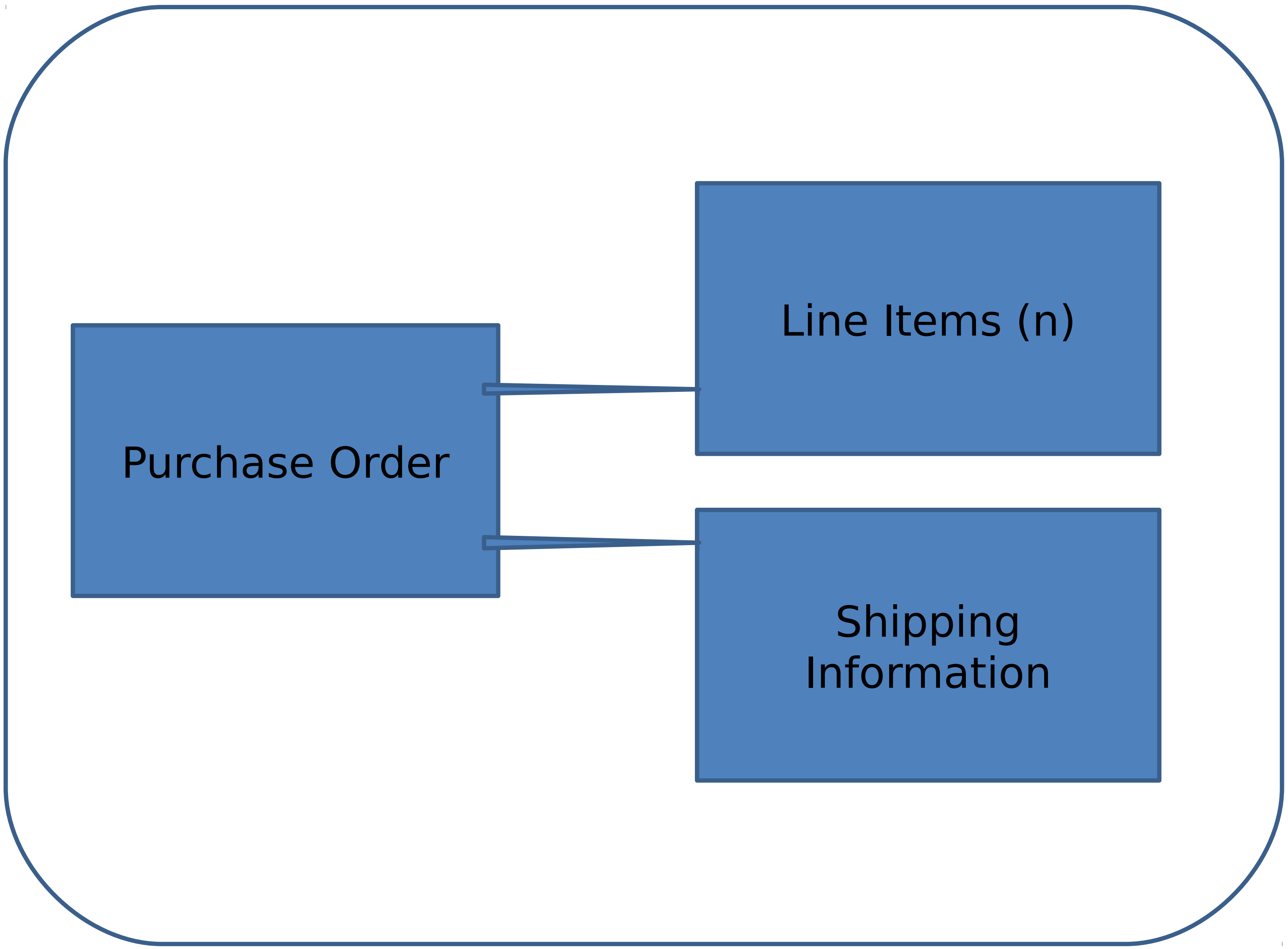
Greg Young Qcon London

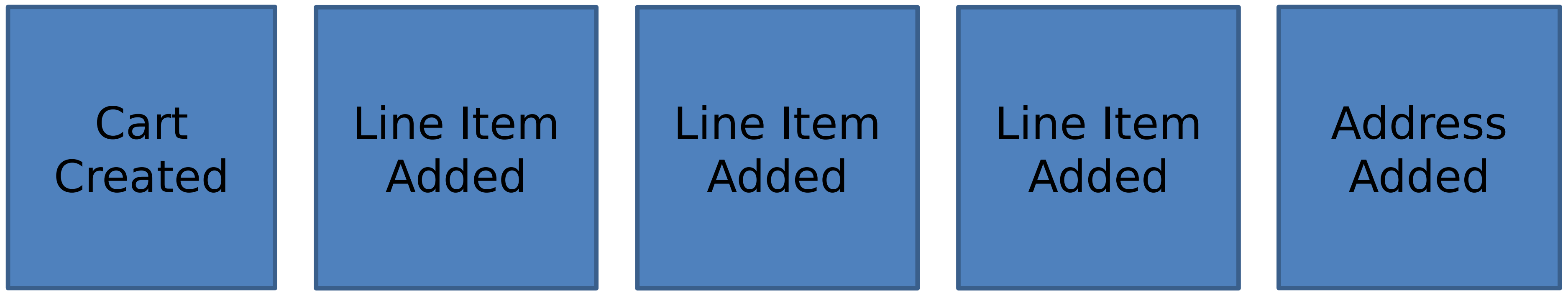
Agenda

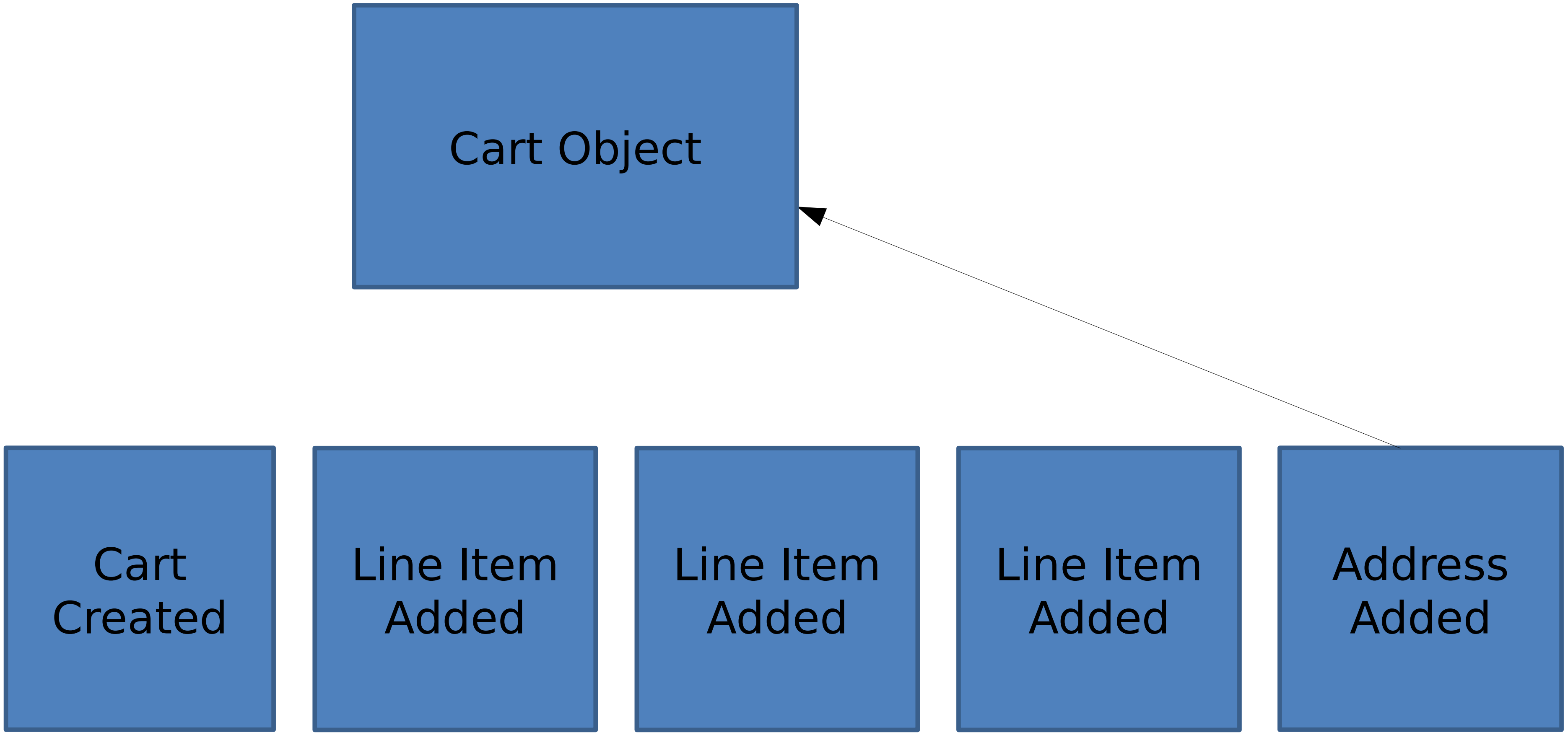
Event Storage
Testing With Events
Versioning
Performance
Traveling Through Space and Time
Let Everyone Run to Airport to Catch Flights

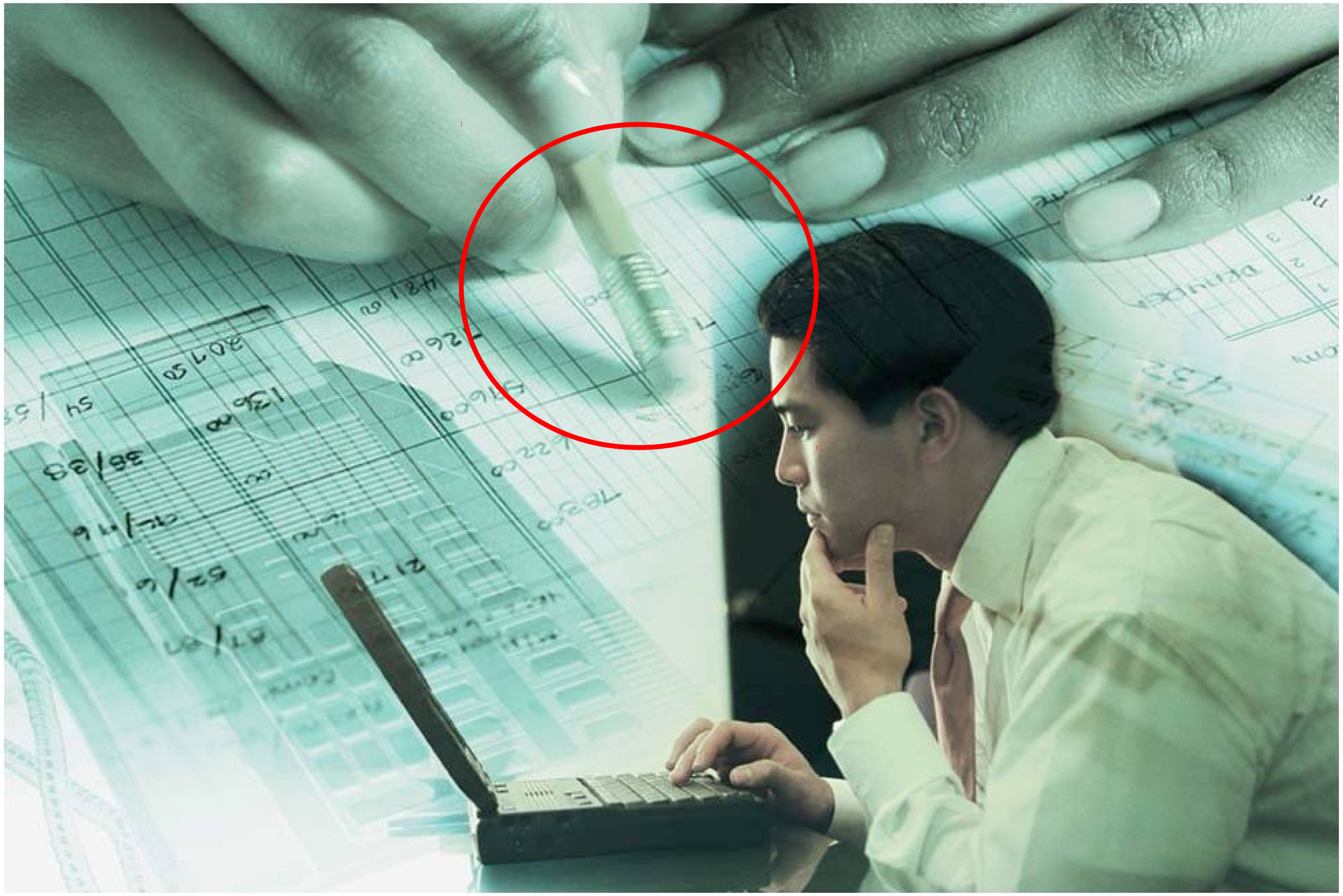

HELP

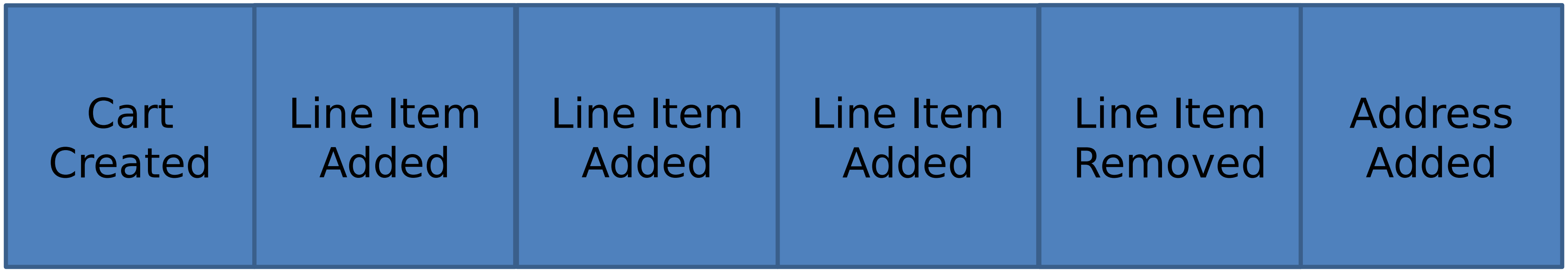

Fines
PK FineID
FK1 CategoryID
FinePerDay
Categories
PK CategoryID
CategoryName
patrons
PK patronID
fname
lname
email
FK1 CategoryID
FK2 AID
Addresses
PK AID
number
street
FK2 city
FK1 zip
Cities
PK CityID
City
ZipCode
PK ZipID
ZIP
FK1 StateID
States
PK StateID
State
Publication_Loans
PK Publication_LoansID
FK1 ISBN_ISSN
FK2 patronID
dateLoaned
dueDate
dateReturned
daysLate
Publication_AuthorJT
PK PAJTID
FK1 authorID
FK2 ISBN_ISSN
Authors
PK authorID
author_fname
author_lname
PubType
PK PubTypeID
type1
Publications
PK ISBN_ISSN
FK1 publisherID
FK2 titleID
date_of_pub
FK3 PubTypeID
Publishers
PK publisherID
publisherName
Titles
PK titleID
title
Publication_Copies
PK Publication_CopiesID
FK1 ISBN_ISSN
numberOfCopies
Publication_SubjectsJT
PK PSJTID
FK1 subjectID
FK2 ISBN_ISSN
Subjects
PK subjectID
subjectName

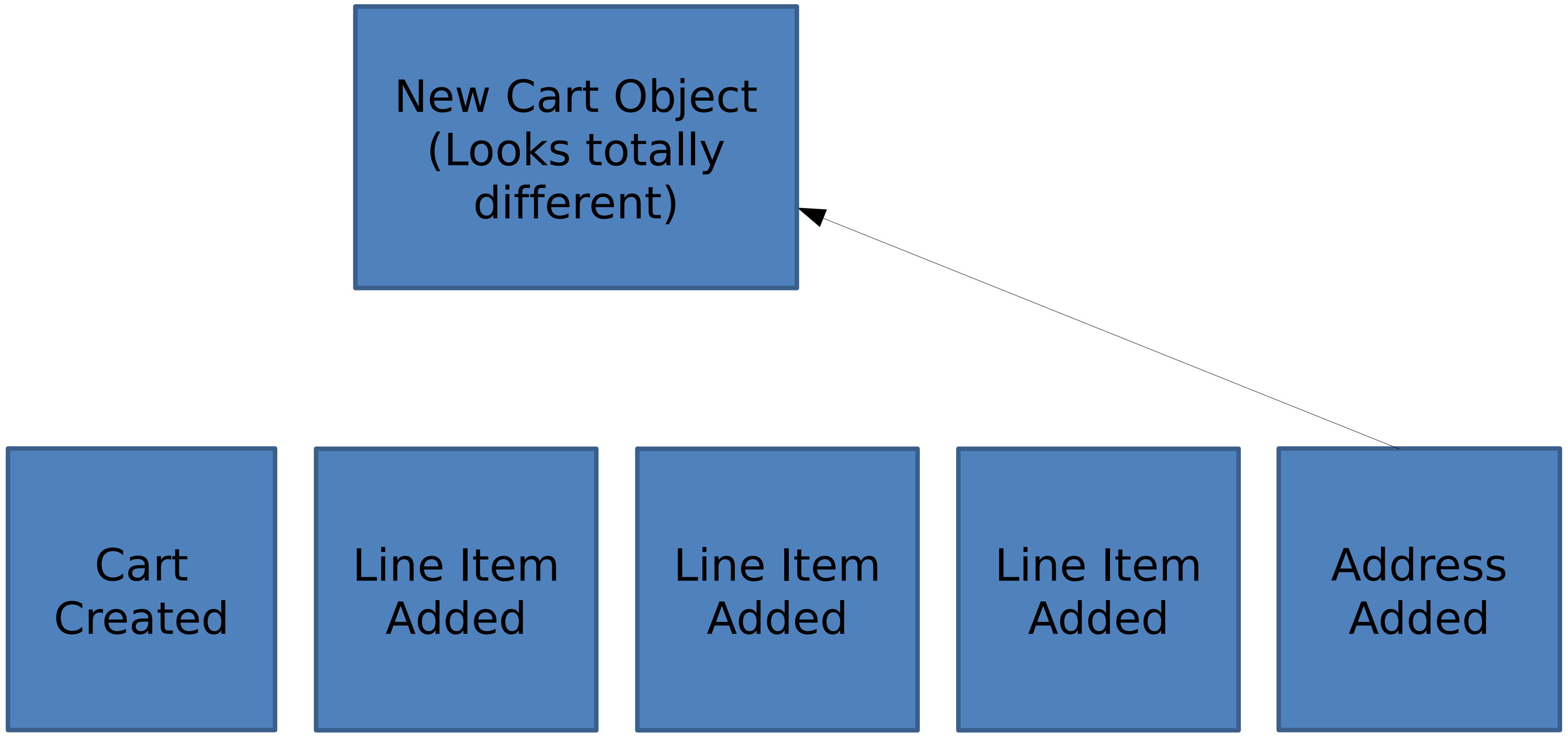

Purchase Order
Line Items (n)
Shipping Information

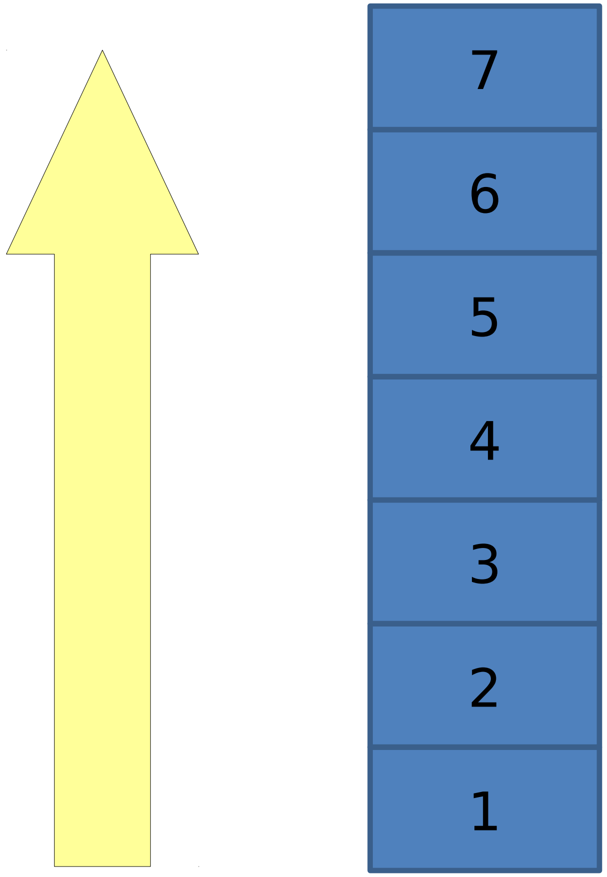

Cart Created
Line Item Added
Line Item Added
Line Item Added
Address Added

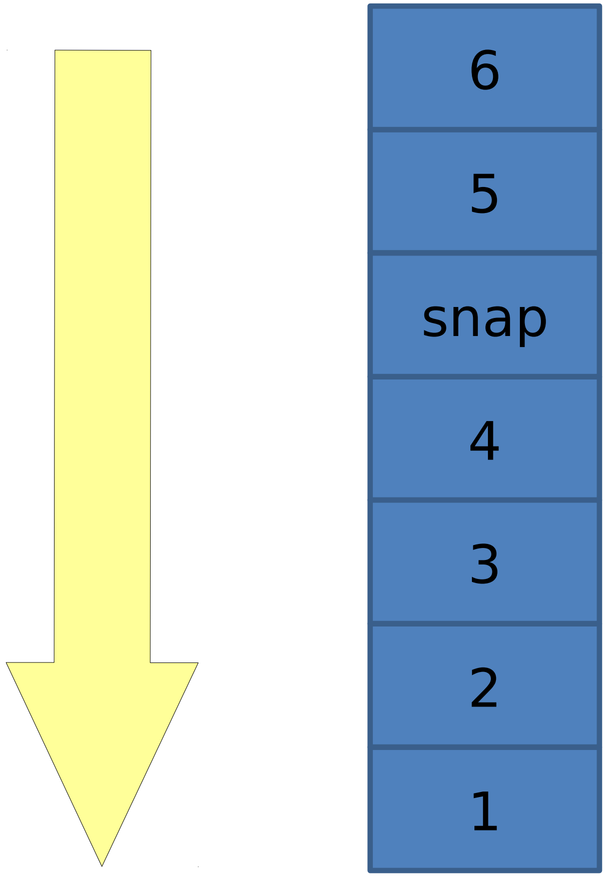

Cart Object
Cart Created
Line Item Added
Line Item Added
Line Item Added
Address Added




Cart Created
Line Item Added
Line Item Added
Line Item Added
Line Item Removed
Address Added


New Cart Object (Looks totally different)
Cart Created
Line Item Added
Line Item Added
Line Item Added
Address Added


7
6
5
4
3
2
1


6
5
snap
4
3
2
1

```
Base {
      public IEnumerable<Event> GetUncommittedChanges()
       {
           return _changes;
       }

       public void MarkChangesAsCommitted()
       {
           _changes.Clear();
       }

       public void LoadsFromHistory(IEnumerable<Event> history)
       {
           foreach (var e in history) ApplyChange(e, false);
       }
}
```

```
Base {
        protected void ApplyChange(Event @event)
        {
            ApplyChange(@event, true);
        }

        private void ApplyChange(Event @event, bool isNew)
        {
            this.AsDynamic().Apply(@event);
            if(isNew) _changes.Add(@event);
        }
}
```

```
        private void Apply(InventoryItemDeactivated e)
        {
            _activated = false;
        }

        public void Deactivate()
        {
            if(!_activated) throw new 
InvalidOperationException("already deactivated");
            ApplyChange(new InventoryItemDeactivated(_id));
        }
```

```
Derived {
        protected void ApplyChange(Event @event)
        {
            ApplyChange(@event, true);
        }

        private void ApplyChange(Event @event, bool isNew)
        {
            this.AsDynamic().Apply(@event);
            if(isNew) _changes.Add(@event);
        }
}
```

Volume
Chart
Value Change

Overdraw attempts are rejected

Given

An account with $100

When

A debit is requested for $101

Then

An InsufficientBalanceException is thrown

Overdraw attempts are rejected

Given

An account was created.
An initial deposit was made for $100

When

A debit is requested for $101

Then

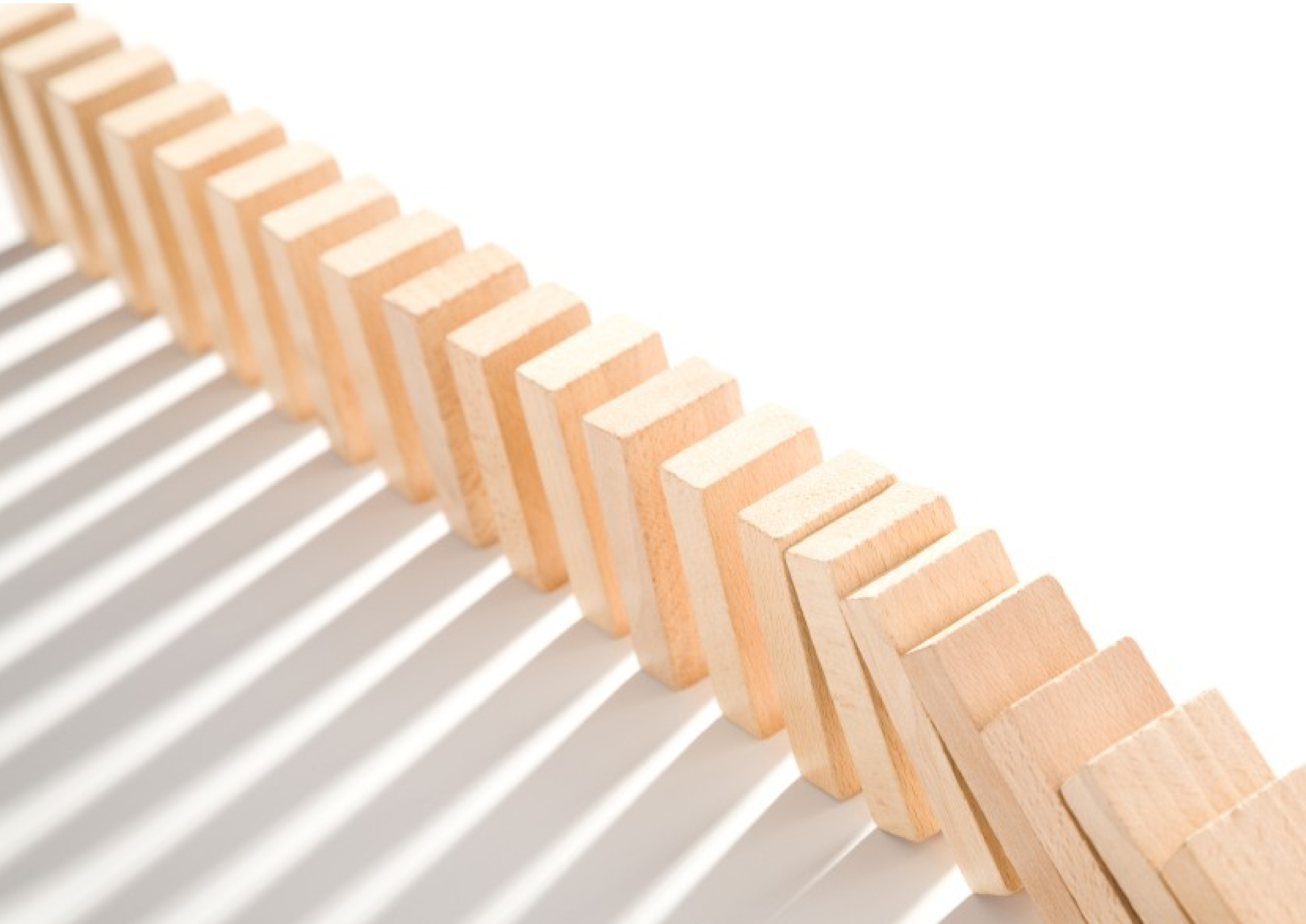
An InsufficientBalanceException is thrown

Overdraw attempts are rejected

Given
A series of events

When
A command

Then
A series of events



Not allowed to take money out of an overdrawn account

Given
An account was created.
An initial deposit was made for $100
$100 was withdrawn

When
A debit is requested for $20

Then
An InsufficientBalanceException is thrown

Traveling Through Time.