# Executable Domain Model

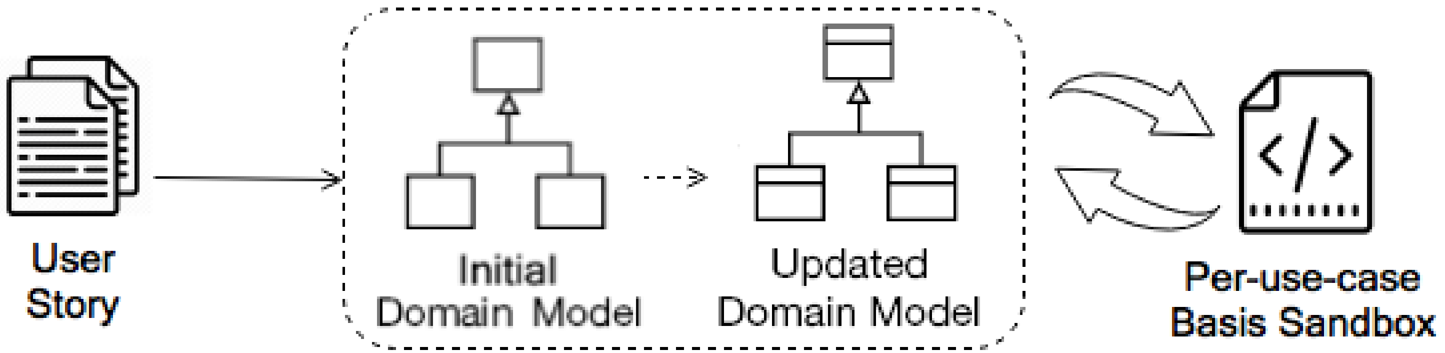
Bo Wang

# Outline

- Domain Models
- Domain-driven and ICSM
- **Current Progress & Infrastructures**
- Future Directions

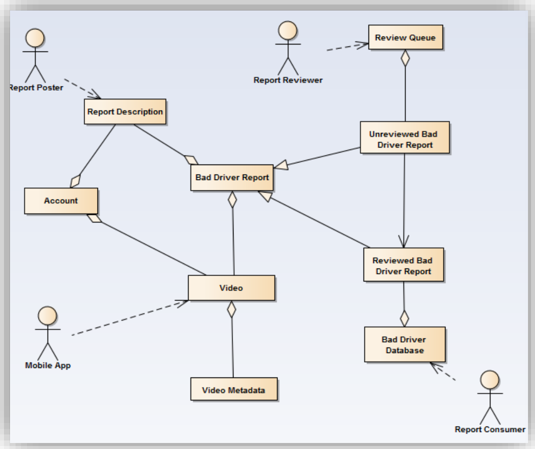
# Domain Models

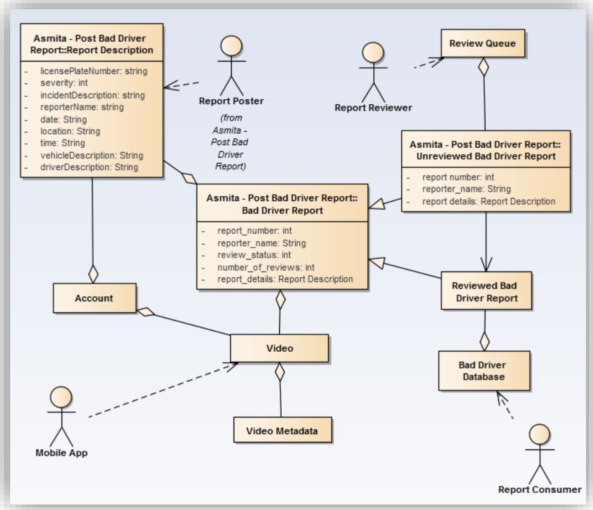
- Operational System Context
- **Application Domain Model**
  - General domain terminologies, basic notions of the application domain

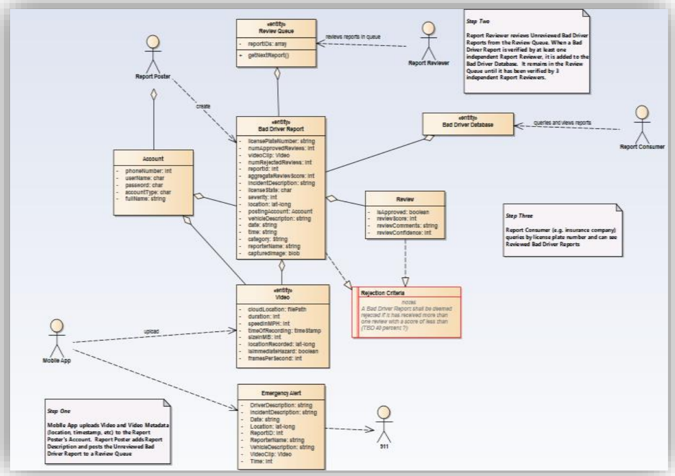


# Domain Models

- Domain Model Evolution




Asmita - Post Bad Driver Report::Report Description
- licensePlateNumber: string
- severity: int
- incidentDescription: string
- reporterName: string
- date: String
- location: String
- time: String
- vehicleDescription: String
- driverDescription: String
Report Poster
(from Asmita - Post Bad Driver Report)
Report Reviewer
Review Queue
Asmita - Post Bad Driver Report:: Unreviewed Bad Driver Report
- report number: int
- reporter_name: String
- report details: Report Description
Asmita - Post Bad Driver Report:: Bad Driver Report
- report_number: int
- reporter_name: String
- review_status: int
- number_of_reviews: int
- report_details: Report Description
Account
Video
Reviewed Bad Driver Report
Bad Driver Database
Mobile App
Video Metadata
Report Consumer



*Step One*

Mobile App uploads Video and Video Metadata (location, timestamp, etc) to the Report Poster's Account. Report Poster adds Report Description and posts the Unreviewed Bad Driver Report to a Review Queue

*Step Two*

Report Reviewer reviews Unreviewed Bad Driver Reports from the Review Queue. When a Bad Driver Report is verified by at least one independent Report Reviewer, it is added to the Bad Driver Database. It remains in the Review Queue until it has been verified by 3 independent Report Reviewers.

*Step Three*

Report Consumer (e.g. insurance company) queries by license plate number and can see Reviewed Bad Driver Reports

**Development Lifecycles**

# Domain-driven and ICSM

- 3 phases focusing on domain modeling

|  |  |  |  |
|---|---|---|---|
| Phase | Inception | Foundations$_1$ | Development$_1$<br>Foundations$_2$ … |
| Activity | • Initial scoping<br>• Concept definition | • Operational concept<br>• System architecture | • Increment design<br>• Increment foundation rebaseline |
| Task | • Initial Domain model **Identification** | • Domain model **prototype & analysis** | • Domain model **update & integration**<br>• Detailed design |

**Project Lifecycles**

- Initial Domain model **Identification**

- Domain model **prototype & analysis**

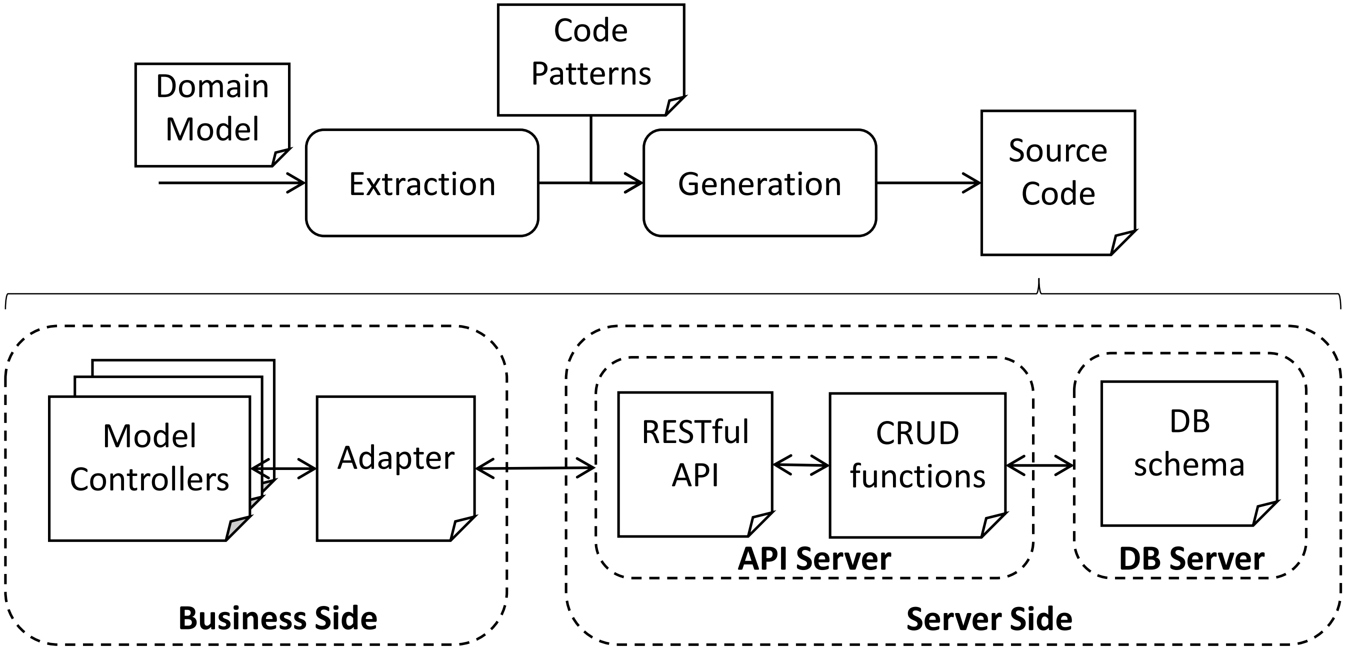
- Domain model **update & integration**
- Detailed design

- Code generation built upon

**Problem Domain Abstraction**

**Easy Access**

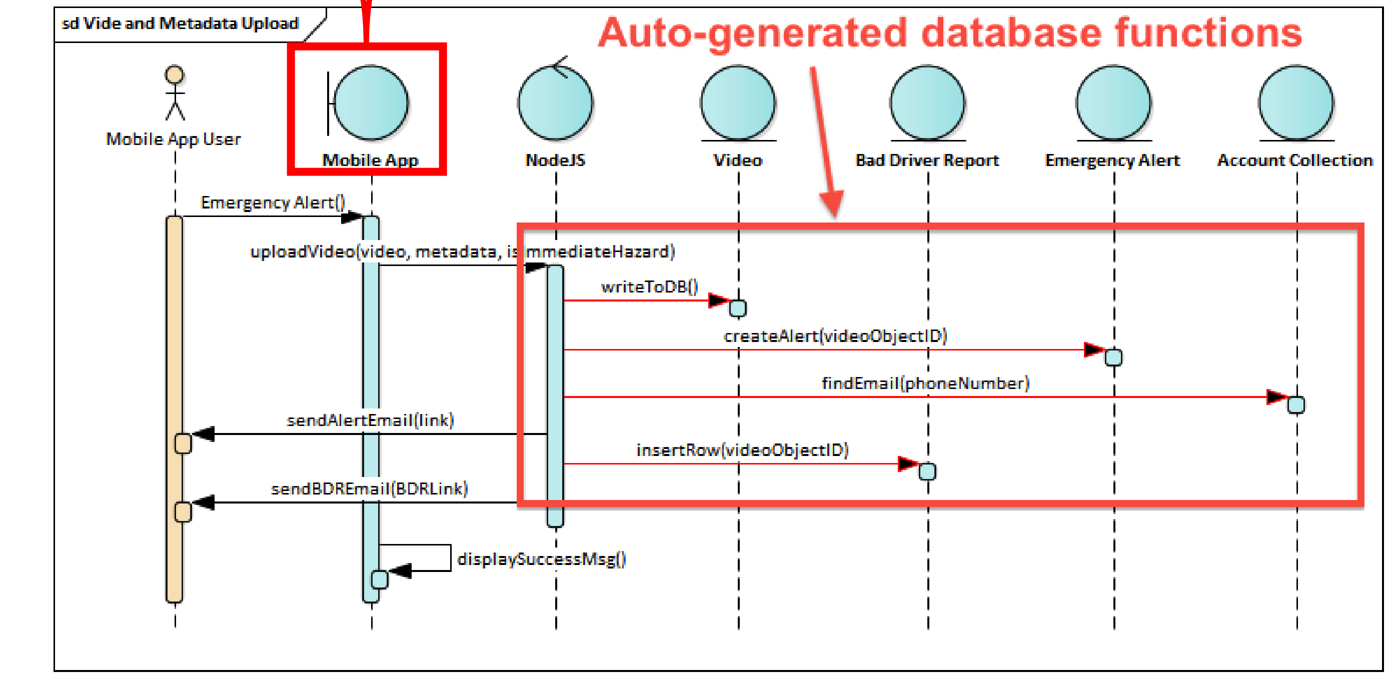
**Flexible Schema**



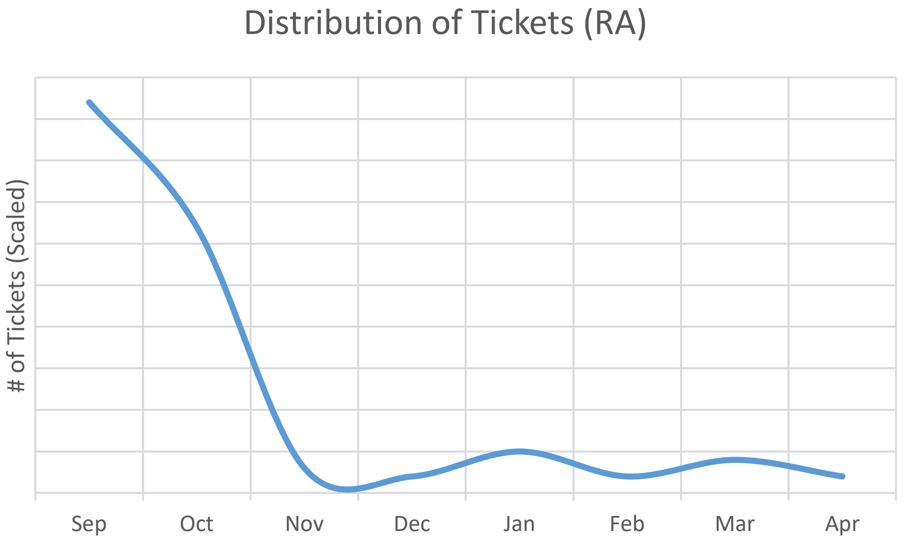
# Overview of generated code and benefits

Java

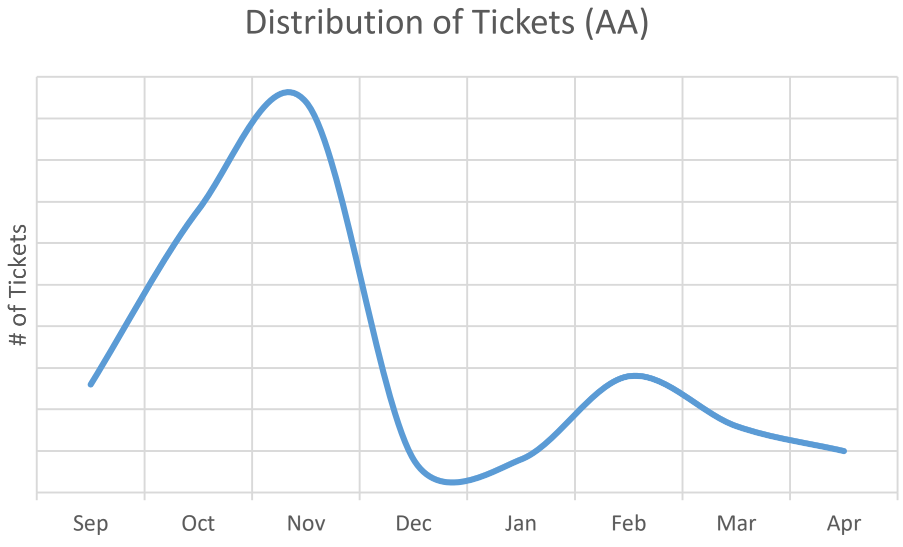
…



# Comparison of tickets distribution



Agile development with Executable Domain Model



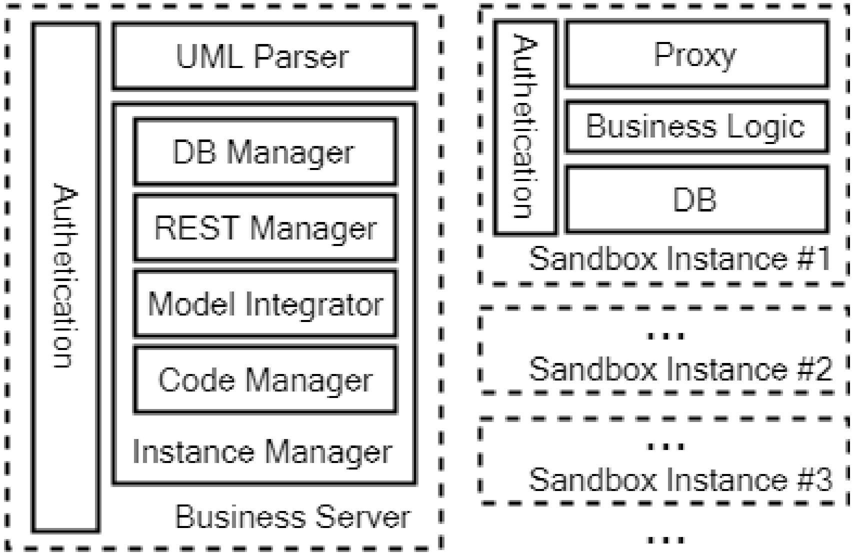
General Agile Development in CS577

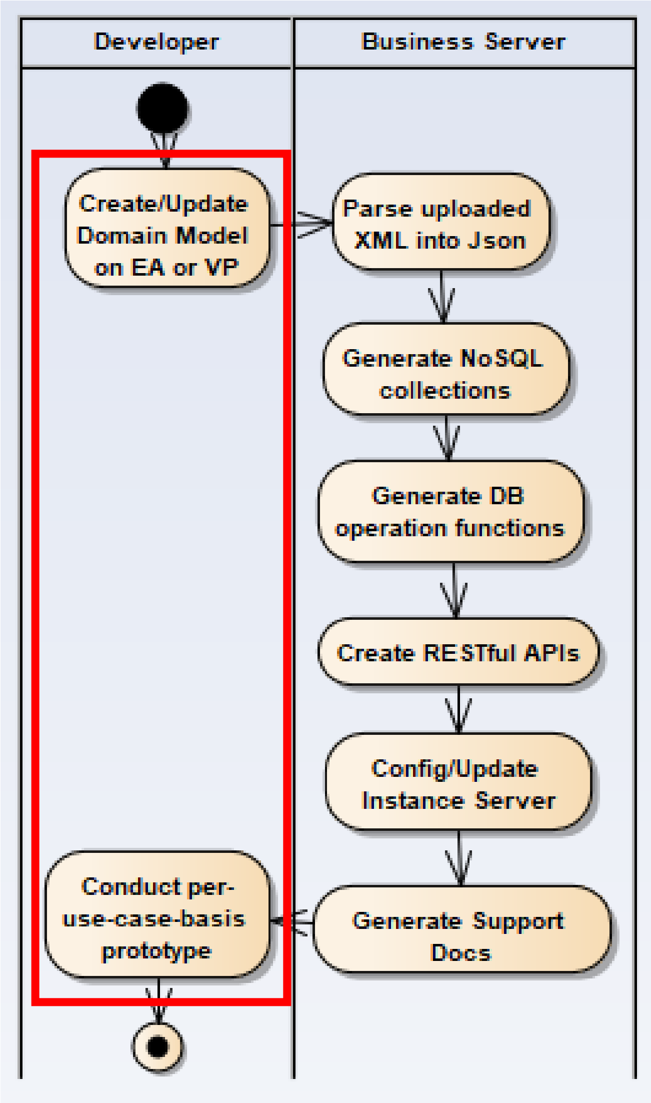
Tickets related to domain modeling and server-side construction for service-oriented projects

- More effort spent on design update in early stages
- Less issues generated in later stages

# Infrastructures

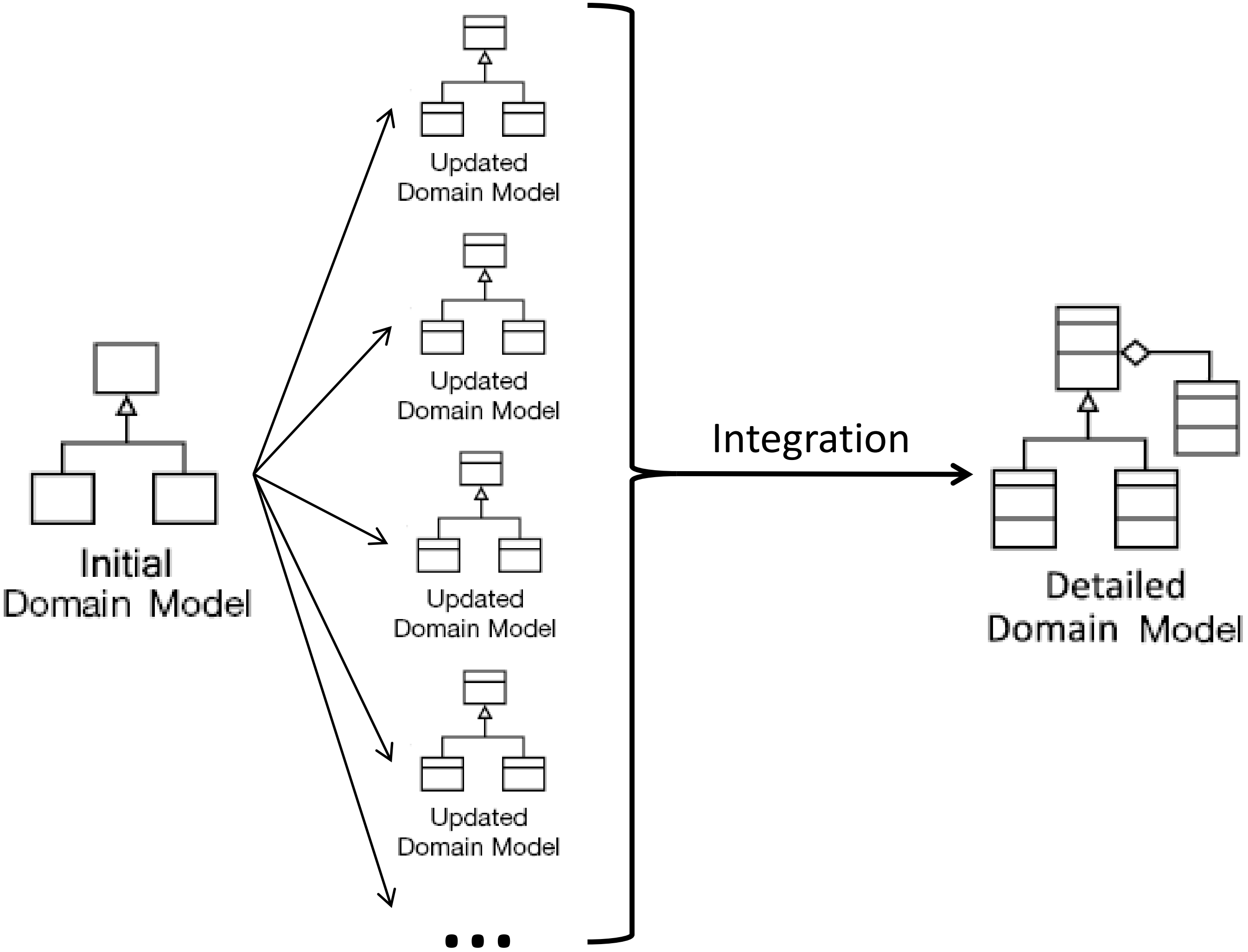
- Domain Modeling Tool
  - —Enterprise Architect (EA) & Visual Paradigm(VP)
- Amazon EC2
  - — Business Server (Python Flask)
  - — RESTful API server (NodeJS), DB (MongoDB)




Developer
Business Server
Create/Update Domain Model on EA or VP
Parse uploaded XML into Json
Generate NoSQL collections
Generate DB operation functions
Create RESTful APIs
Config/Update Instance Server
Generate Support Docs
Conduct per-use-case-basis prototype

# Enable per-use-case-basis sandbox development in parallel

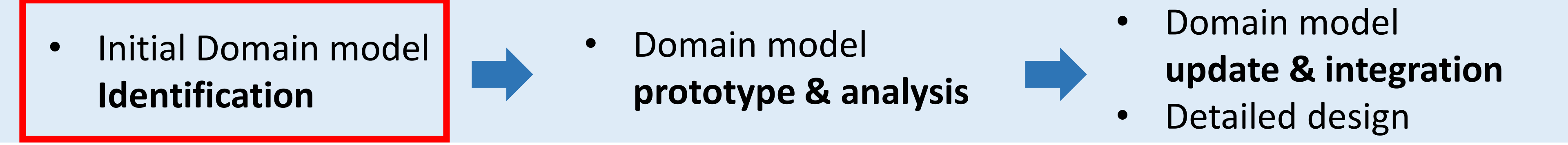
-



# Feedback-driven on bug-free featured templates development

- Functionality
  - Large files, Email, Spatial querying, etc.
- Server-side Observability
  - Logging, Exception, Health
- Security
  - Encryption and Permission control

- Experience gained from Parallel Agile projects
  - original concept (LBA)
  - need for large file support (CarmaCam)
  - need for API documentation (TikiMan and CarmaCam)
  - need for database and API security (CarmaCam)
  - …



- Initial Domain model **Identification**
- Domain model **prototype & analysis**
- Domain model **update & integration**
- Detailed design

- Extract potential entities from User Story
  - ~~—Use case specification~~
  - —Win conditions
    - 39 projects, 865 win conditions
    - 80% **Functional requirements**

      As [a|an] …[, ][I i] [can|should|shall]…
    - 12% non-functional requirements

      (The)?[Ss]ystem [shall|should|must]…
    - 8% Others
- TDs and POS-tags
  - — Type Dependencies (TDs): Grammatical dependency between the words of a sentence.
  - — Parts-of-Speech tags (POS-tags): words in a sentence tagged (or annotated) with parts of speech, such as noun, pronoun, verb, adjective, etc.

As a user , I can view other users' profiles so that I can see their information and points.

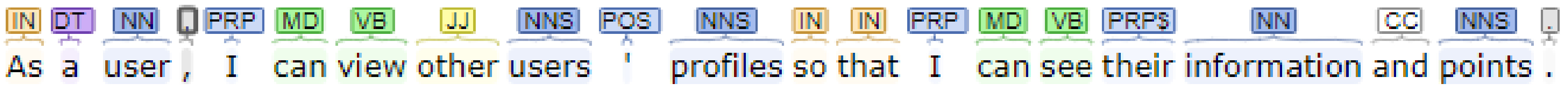

IN DT NN , PRP MD VB JJ NNS POS NNS IN IN PRP MD VB PRP$ NN CC NNS .
As a user , I can view other users ' profiles so that I can see their information and points .

Figure out nouns based on POS-tags

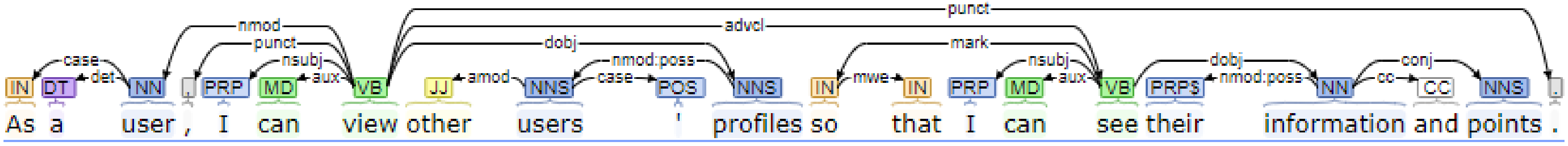
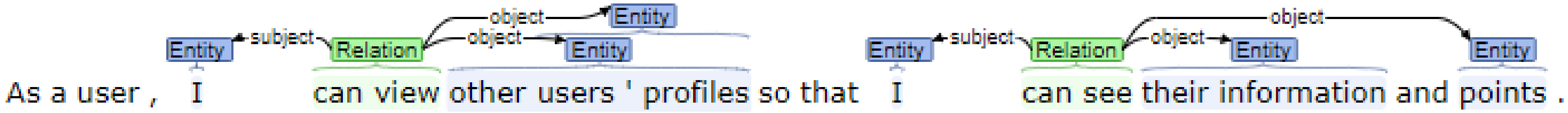

punct
advcl
dobj
mark
nmod
punct
nsubj
case
det
aux
amod
nmod:poss
case
mwe
nsubj
aux
dobj
nmod:poss
conj
cc
IN DT NN , PRP MD VB JJ NNS POS NNS IN IN PRP MD VB PRP$ NN CC NNS .
As a user , I can view other users ' profiles so that I can see their information and points .

Figure out Sentence Structure Rule based on TDs


Entity
subject
Relation
object
Entity
object
Entity
Entity
subject
Relation
object
Entity
object
Entity
As a user , I can view other users ' profiles so that I can see their information and points .

Users’ Profiles

- information
- points

- **Intersection**

  “email” appears in both user case.

- **Intersection***

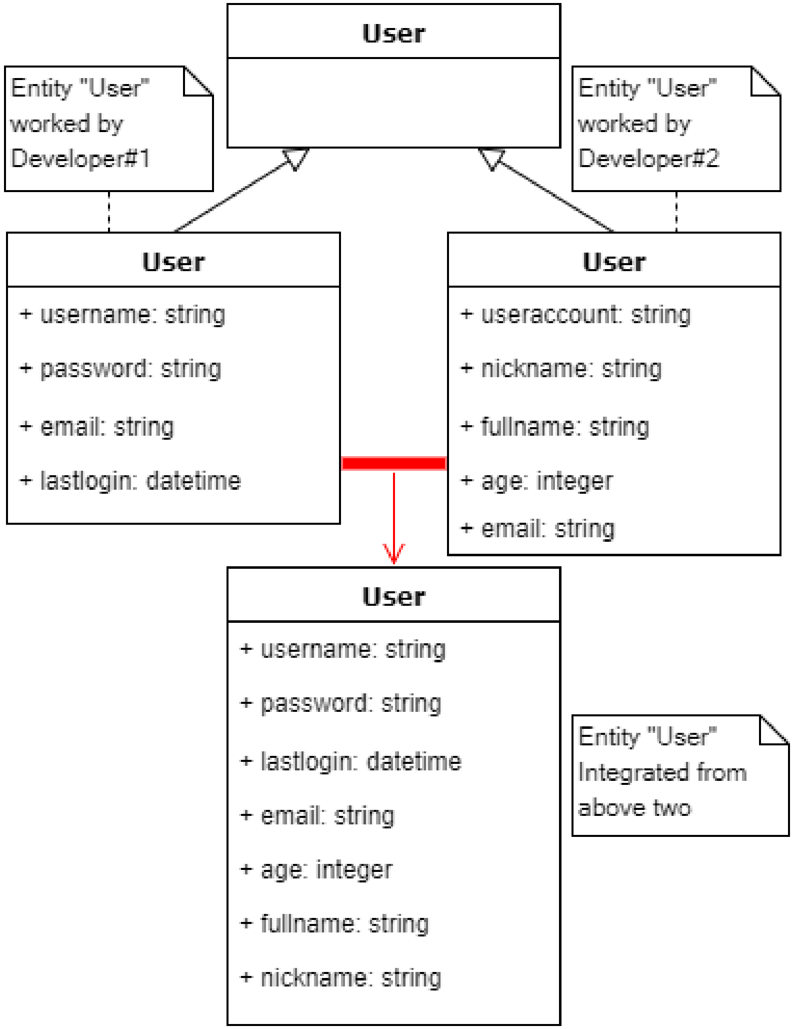
  “username” and “useraccount” in different use case are talking about the same thing.

- **Relative Complement**

  “password” and “lastlogin” only appear in Developer#1’s use case while “nickname”, “fullname” and “age” only appear in Developer#2’s use case.

- There may be other cases (TBD)

  E.g. an attribute is removed or moved from User entity to another entity. (Out of the problem scope)



# Solution

- Heuristic:
  - —Infer from the DataType of attribute. It will partition potential attributes with same data type. Attributes with different data type cannot be **Intersection***
  - —Infer from the testing data about the value format of the attributes. Same data format (e.g. email format, phone # format, all digits, same length or same complex data type, etc.) may denote to **Intersection***
  - —Semantic Similarity could be cost function:
    - Pre-define a synonyms model, then figure out Cosine or Euclidean during the runtime.
- The result could be sent to human for final adjustment

# Thank You

- Questions?