DDD at 10

Eric Evans
domainlanguage.com
@ericevans0 #dddesign

Domain-Driven Design (DDD)

- Focus on the *core domain*.
- Explore *models* in a creative collaboration of software practitioners and domain practitioners.
- Speak a *ubiquitous language* within an explicitly *bounded context*.

DDD didn’t start in 2003

… Design by Contract, Bertrand Meyer Responsibility-Driven Design, Rebecca Wirfs-Brock, Ward Cunningham, Kent Beck, Smalltalk, Dave Thomas, Ableson & Sussman, Grady Booch, Ralph Johnson, Eric Gold, David Siegel …

First cycle 2003-2007

Points I had felt I had to make

- Modeling (to be useful) must be linked to implementation

- Implementation had to be precise and of a different style than mainstream code.

Would not come across without detailed, concrete examples → Deep and long on:

- Building Blocks

Points I also made

- Models emerged through iteration and collaboration with domain experts. (Stated early and throughout.)

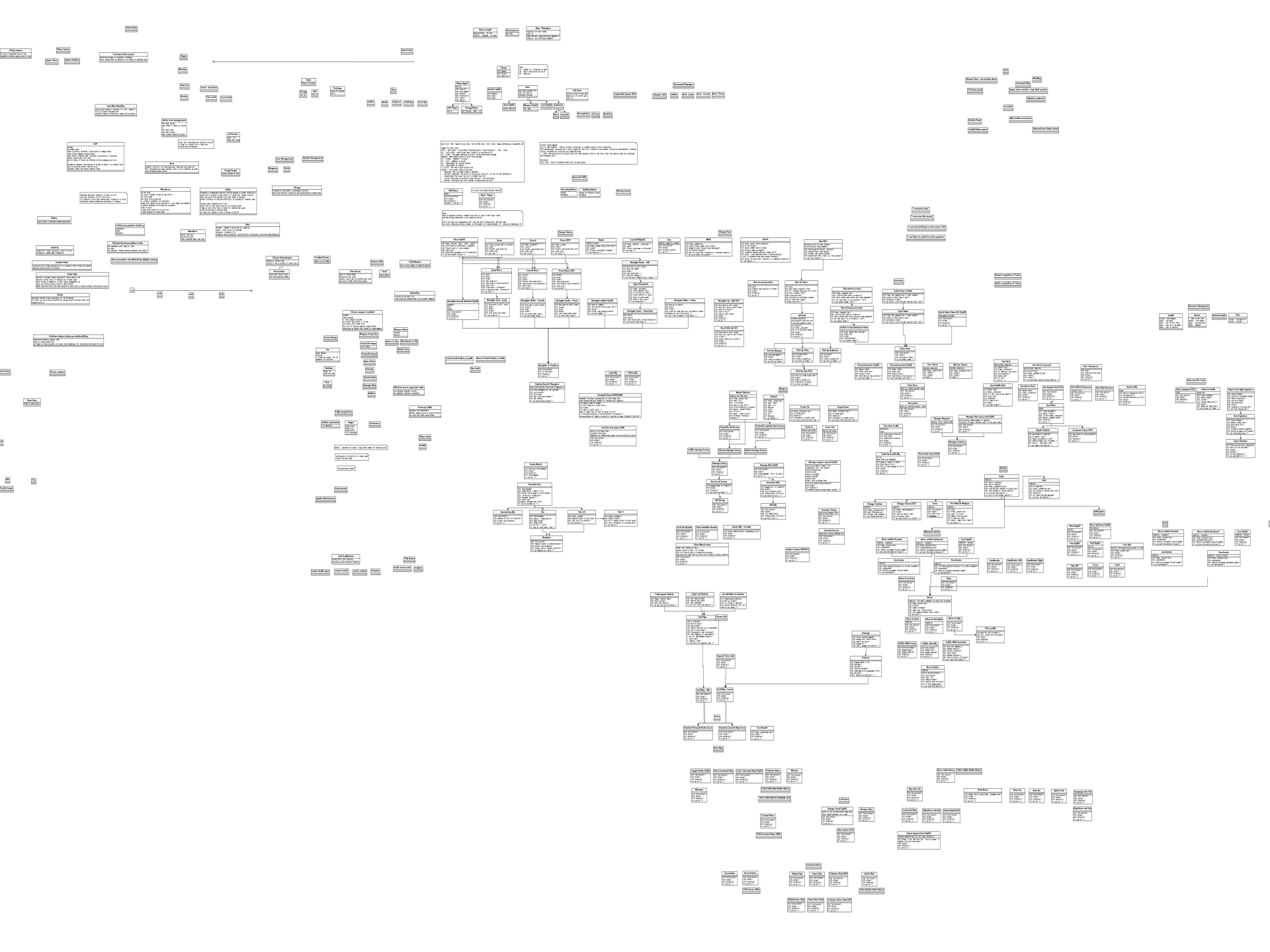
- Certain strategic considerations make success with modeling possible/impossible. (Late in the book)

Top questions of this era

- What's the difference between an entity and a value object?
- Can an entity refer to a repository?
- How do I …. Repository ….?
- Can I … Repository … ?
- Repository … repository … ?

Distressing Trends

- Generate code from UML diagrams.
- Heavy frameworks that intruded into object design and made it difficult to express a model in code, and any OO implementation heavy, heavy.
- Assumption that I would be in favor of <fill in heavy, diagram-centric, modelers don't design approach>



Many Bright Spots

- Jimmy Nilsson wrote a book (2006)

Model, a system of abstractions

- **Not** UML diagram (helps document or communicate parts of some types of models)
- **Not** a layer of the software (helps us make software that behaves consistently with a particular model, and, ideally, expresses the concepts explicitly)
- **NOT** a comprehensive schema for all data needed in a system!

Model, a system of abstractions

- *Abstraction* means leaving stuff out!

Model, a system of abstractions

- Models are not real; Domains are “real”.
- Even real*ism* is a distraction in modeling; A model should allow us to make assertions.
- A model might be predictive; A model might isolate factors leading to decisions.

Bounded Contexts

- An explicitly defined part of a software system in which a model's definitions and assertions strictly apply.

- Typical boundaries: a subsystem, a service boundary, a set of packages and a set of tables … an unusual technology platform

Strategic Design emphasis (2005 -)

No more idealistic
legacy replacement projects please!

Isolate something strategic (core domain)
in a fresh bounded context, and ***focus***!

Typical Question of 2007

- DDD says you must have an O-R mapper and layered architecture. Correct?

2008-2011

Event Sourcing and CQRS

Event Sourcing

- Greg Young

- No more mutable objects!
- State changes through the accumulation of Domain Events

Domain Events

- A representation of something that happened in the domain, or
- A rule-based response to one or more other domain events.

- New building block pattern (existing ones sharpened)

CQRS

- Udi Dahan

- Separate processing new information or commands from viewing/analyzing the state.
- Break the monolith into focused chunks joined (mostly) by domain events.
- Emphasis on bounded contexts.

DDD Architectures for Volume and Distribution

- Some confusion of definitions.
- Breaking the logjam takes a certain kind of personality.
- Common Question: Are these things DDD?
  - Yes they are.

DDD with Agile (2009 -)

No explicit place for design/
Naive notions of emergent design.

Get your story done!
Keep up the velocity!!
Not fun anymore.

Modeling and design are often the quickest path to the goal. What is the actual goal?

(Hint: Not a user story.)

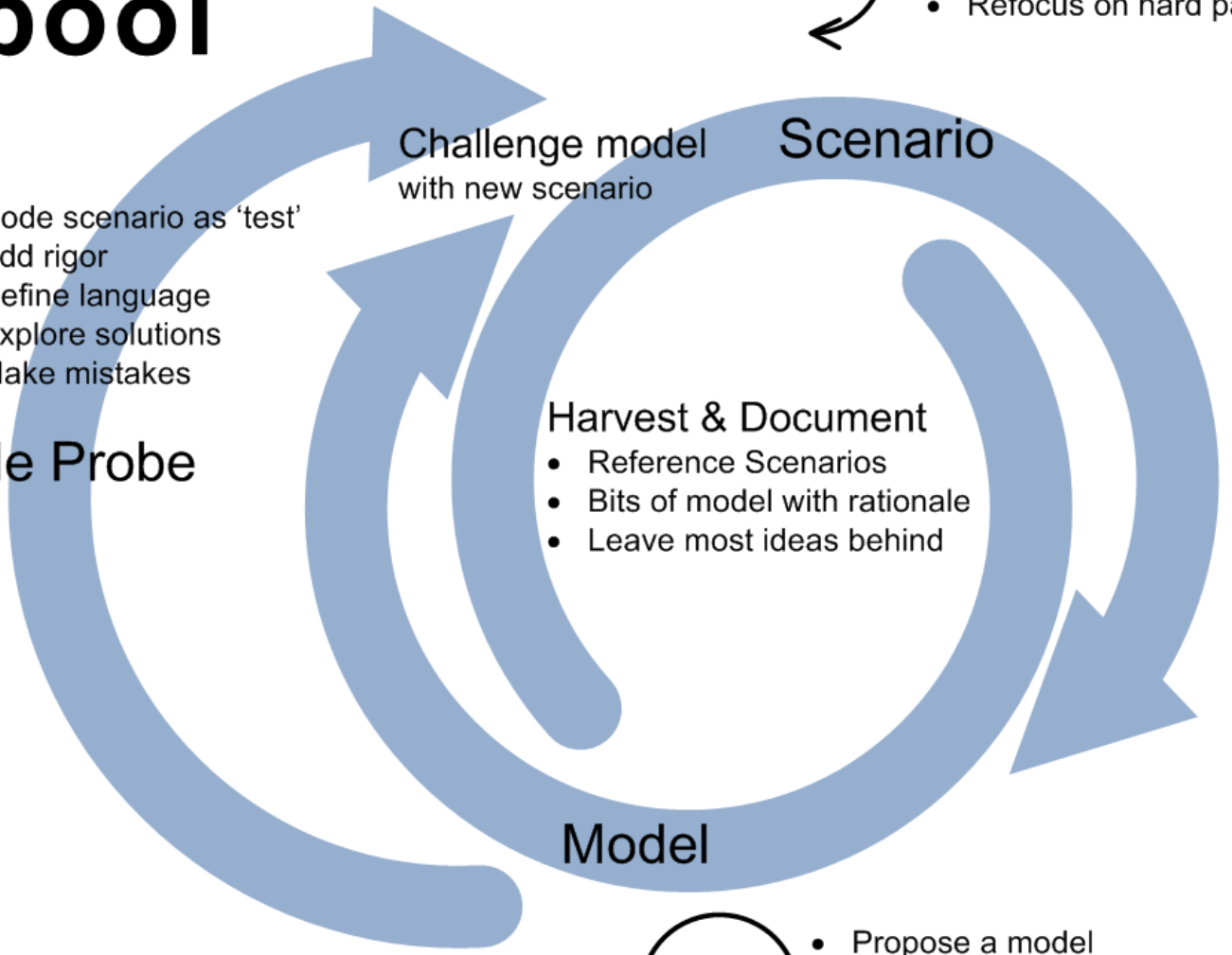
Boldly explore the domain and models of the domain!

We don’t know the future.

Model Exploration

Whirlpool

DRAFT 0.2



- Tell us a story.
- Flesh it out.
- Refocus on hard part.

- Code scenario as 'test'
- Add rigor
- Refine language
- Explore solutions
- Make mistakes

- Research
- Contemplation

- Propose a model
- Walkthrough states
- Walkthrough solutions
- Explore language
- Make mistakes

Bubble Contexts

- No more long-term tradeoff.
- Intensive modeling in the core domain. **Now!**
- Short-lived bounded contexts with tentacled anticorruption layers.
- Show what modeling and design can do when we don’t drag along the mundane parts.

Winds Shifting Toward Design

Client-server

- Fat client / thin client … It's all monolithic
- Architectural/technical monoculture

Distribution, concurrency, eventual consistency…

- big rewards for small, loosely-coupled modules
- big rewards for clear assertions

Open-source let people *show* their design work.

2011-

Not *only* SQL

Not *only* Objects

NoSQL

- Light-weight, fast ways to manage data.
- Alternative modeling paradigms.

Model, a system of abstractions

We now have three or five
viable modeling paradigms.

… complete with implementation
platforms and communities.

Object Oriented

- First to explicitly focus on domain modeling
- Units of abstraction: classes, methods, references …
- Behavioral aspects of model procedural, encapsulated in methods
- Original building block patterns reflected what we had learned about how to shape a model that could be crisply expressed in OO.

Event Sourcing

- Sequence of events
- Projection to produce object representing state.
- Projected objects typically made of entities and value-objects
- Projection typically computed by objects

Relational

- Not the comprehensive normalized schema.
- (Abstraction means leaving stuff out.)

- A weak form of relational algebra

- Set operations

Functional

- Units of abstraction: functions, sequences, maps …
- Building blocks? (Values, yes. Entities, altered. Domain Events …)
- Event sourcing?

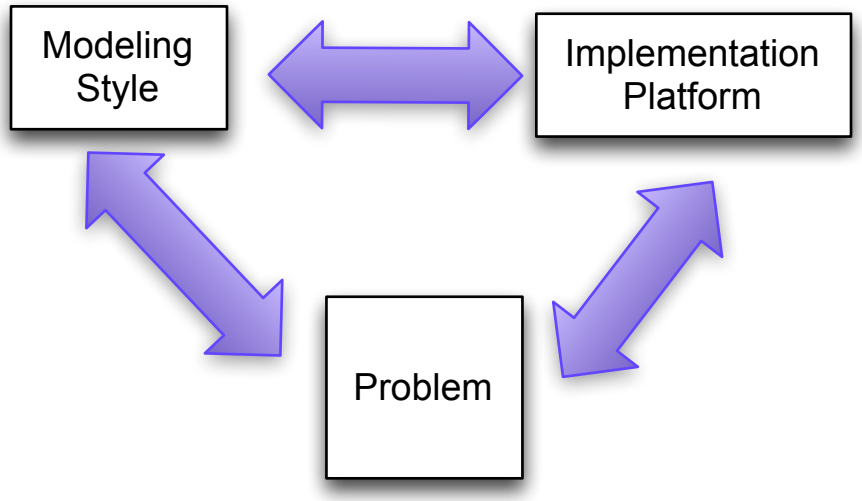
Graphs

- Units of abstraction: Nodes, edges, …
- Fuzzy, approximate
- Social

Logic / Rules Engines

- Still haven't seen it. Keep hoping.

Modeling in Other Paradigms



- Yes, the *problems we solve* are influenced by the modeling style and platform. Don't kid yourself.

Domain-Driven Design (DDD)

- Focus on the *core domain*.
- Explore *models* in a creative collaboration of software practitioners and domain practitioners.
- Speak a *ubiquitous language* within an explicitly *bounded context*.

DDD at 10

Eric Evans
domainlanguage.com
@ericevans0 #dddesign