NTCODING.CO.UK

2017

THE STRATEGIC PRACTICES OF DOMAIN-DRIVEN DESIGN

SUPPORTING MATERIAL FOR NICK TUNE'S ADVANCED STRATEGIC DDD WORKSHOP

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN

THE STRATEGIC PRACTICES OF DDD
BY NICK TUNE (@NTCODING)

Nick Tune is an experienced technical leader. He has helped teams in a variety of organisations to achieve continuous delivery and high alignment, including the UK government, Salesforce, and 7digital. He is the co-author of Patterns, Principles and Practices of Domain-Driven Design, a regular conference speaker, and a workshop enthusiast. Above all, a lifelong student who loves sharing as he learns.

Website:	ntcoding.co.uk
Blog:	ntcoding.co.uk/blog
Talks:	ntcoding.co.uk/speaking
Twitter:	@ntcoding
LinkedIn:	linkedin.com/in/ntcoding

You can download this booklet from
ntcoding.co.uk/workshops/strategic-ddd-practices

TABLE OF CONTENTS

WHAT IS DOMAIN-DRIVEN DESIGN?

There's an alignment crisis in software development, but Domain-Driven Design empowers software developers to solve it.

As an industry we are resigned to many kinds of failure happening on software projects; developers building the wrong feature, low value parts of the system being gold plated while high value areas are neglected, or teams constantly slowing each other down. These are all symptoms of low alignment: our organisational and technical designs do not align with the problem domain.

Important information is lost in translation from the business world to the technical world leading to poor decision making and missed opportunity.

No magical solutions exist to instantly make the pains of low alignment disappear. But Domain-Driven Design encourages a mindset of improving alignment, enabling DDD practitioners to minimise the costs of translation and fix some of the most important business, organisational, and technical challenges in their organisations. If you adopt the DDD mindset, you will learn how to solve these problems in your organisation, regardless of your job title or seniority.

DDD encourages alignment at all levels; Aligning with the business vision to apply engineering effort where payback is greatest. Aligning the organisation with the domain to create autonomous teams. Aligning technical architecture with the domain, so autonomous teams have full ownership of their code.

DDD practitioners also align their code with the domain, ensuring the code model uses the same words as domain experts speak to keep costs of translation to an absolute minimum. The coding patterns are referred to as tactical DDD. But they provide little benefit when the strategy is flawed.

LEARN MORE

- **Domain-Driven Design** [book] - https://www.amazon.com/Domain-driven-Design-Tackling-Complexity-Software/dp/0321125215
- **The Anatomy Of Domain-Driven Design** [booklet] - https://leanpub.com/theanatomyofdomain-drivendesign
- **The Anatomy of Domain-Driven Design** [video] https://www.youtube.com/watch?v=nKI5tioa7pg&list=PLIHVWAtJFACm4sffK5M_UxDg7k-xXI16F
- **Strategic and Collaborative Domain-Driven Design** [talk] - http://ntcoding.co.uk/speaking/talks/strategic-collaborative-domain-driven-design

BUSINESS GLOSSARY

Creating high alignment is the key to minimising the translation costs between the problem and solution space. One of Eric Evans' most significant contributions in this area is his obsession with creating consistency in all forms of communication by having everyone involved in a project use a shared vocabulary - the ubiquitous language.

To improve uptake of ubiquitous language in your organisation, you should create a business glossary.

A business glossary is a list of key phrases, their definition and the contexts in which they apply. The goal is for your business glossary to become a shared document used and maintained by the whole team.

As systems and organisations scale it is recommended practice to have a business glossary in each context (contexts are introduced later). It's important to recognise the same term or phrase can have different definitions in different contexts. This is why context boundaries are often linguistic boundaries in DDD.

AN EXAMPLE BUSINESS GLOSSARY

The following is a small extract of a business glossary taken from the Tax Calculation context in a government department.

Term	Definition
Business Property Tax Bill	A single annual statement sent to Ratepayers indicating their business property tax liability for the previous year for all hereditaments
Ratepayer	Every organisation must declare a Ratepayer – the person responsible for ensuring the Business Property Tax Bill is paid

TIPS FOR CREATING A BUSINESS GLOSSARY

- Involve a mix of business, domain, and technical experts
- Treat it as an evolving document where terms are added and amended as your knowledge of the problem domain increases
- Each bounded context has its own ubiquitous language
- Often phrases have multiple meanings in multiple contexts
- Try to find out if there is already an existing business glossary in your organisation

LEARN MORE

- **Eric Evans: What I've learned about DDD since the book** [talk] - https://www.youtube.com/watch?v=lE6Hxz4yomA
- **The Anatomy of Domain-Driven Design Animations** [videos] - https://www.youtube.com/watch?v=nKl5tioa7pg&list=PLIHVWAtJFACm4sffK5M_UxDg7k-xXI16F

THE STRATEGIC PRACTICES OF DDD
BY NICK TUNE (@NTCODING)

DISCOVERING THE MISSION

COLLABORATIVE PRACTICES FOR IDENTIFYING AND COMMUNICATING THE PURPOSE BEHIND THE SOFTWARE SYSTEMS YOU BUILD

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN

MISSION STATEMENT

A mission statement is a short, high-level description of the purpose of an organisation. Mission statements help organisations to focus on what is most important to them. Equally, a mission statement helps DDD practitioners to focus their efforts on areas of the problem domain most relevant to the business.

The role of DDD practitioners is to find their organisation's missions statement rather than to create it. However, it is good practice to help organisations that do not have a mission statement to create one.

AN EXAMPLE MISSION STATEMENT

In 2014, Amazon had the following mission statement:

To be Earth's most customer-centric company, where customers can find and discover anything they might want to buy online, and endeavours to offer its customers the lowest possible prices.

Harley-Davidson has the mission statement:

We fulfil dreams through the experience of motorcycling, by providing to motorcyclists and to the general public an expanding line of motorcycles and branded products and services in selected market segments.

TIPS FOR CREATING A MISSION STATEMENT

- Include what the organisation does
- Include who the organisation serves
- Mention the value the organisation provides
- Don't write any code until you have seen your organisation's mission statement

LEARN MORE

- **Answer 4 Questions to Get a Great Mission Statement** - https://www.forbes.com/sites/patrickhull/2013/01/10/answer-4-questions-to-get-a-great-mission-statement/#32be866567f5

- **Fortune 500 Mission Statements** - https://www.missionstatements.com/fortune_500_mission_statements.html

BUSINESS MODEL CANVAS

Software systems are built to solve business problems and exploit market opportunities. Engineering teams who understand the business model behind the product they are building can make better day-to-day and strategic decisions aligned with their organisation's business model.

Use the Business Model Canvas to map out the key components of your organisation's business model. Use the canvas to create a strong shared understanding between business, domain, and technical experts leading to high alignment between the problem space and the solution space.

A Business Model Canvas is a visualisation of the most important components of a business model (explained below). It was invented by Alexander Osterwalder and Yves Pigneur, and presented in their best-selling book Business Model Generation.

A Business Model Canvas is composed of the following elements:

Customer Segments
Who are the different types of customers the business creates value for? e.g. mass market, software developers, healthcare professionals.

Value Propositions
How does the business create value for its customers? e.g. “Bespoke software development”, “Fun multiplayer video games”. Can be different value propositions for each customer segment.

Channels
How do you communicate with potential customers to promote, sell and deliver your value propositions? e.g. website, physical stores, email.

Customer Relationships
How do you communicate with customers for after-sales issues? e.g. dedicated account management, electronic helpdesk.

Key Activities
What activities are essential to making the business model work? e.g. curating content, responding to market trends, creating recommendations.

Key Resources
What assets does the business need for the business model to work? e.g. intellectual property, staff, technical equipment.

Key Partnerships
Which partners play a significant role in the success of the business model? e.g. restaurants (for a dining reservation site), technology suppliers, outsourcing agencies.

Cost Structure
What are the business' major cost drivers? How are they linked to revenue? e.g. salaries, procurement, licensing, hosting

Revenue Streams
How does the business generate revenue from its value propositions? e.g. transaction fees, recurring membership fees, subscription fees.

AN EXAMPLE BUSINESS MODEL CANVAS

The following Business Model Canvas illustrates the business model of a high-profile, low-cost, package holiday travel agent.

KEY PARTNERS

- Resorts that provide exclusive offers
- Other resorts
- Travel Sys (provides flights and generates packages)
- Bed banks (provide information about each resort: description, images, etc.)

KEY ACTIVITIES

- Sending marketing emails
- Creating recommendations
- Securing exclusive deals
- Packaging

KEY RESOURCES

- Brand
- Discoverability
- Low-cost offers
- Exclusive offers

VALUE PROPOSITIONS

- Cheap holiday package deals to worldwide locations
- Exclusive low cost deals for many resorts

CUSTOMER RELATIONSHIPS

- Electronic helpdesk
- Telephone support

CHANNELS

- Website
- Email
- Affiliate network
- Telephone marketing

CUSTOMER SEGMENTS

- Bargain hunters (no frills holiday seekers living in the UK who want cheap and hassle free)

COST STRUCTURE

- Exclusive offer acquisition costs
- Development and maintenance of packaging service and API
- IT Infrastructure
- Marketing and tech salaries

REVENUE STREAMS

- Commission on holiday bookings
- Website advertisements
- Cross-selling commission (hire cars, guided tours, etc.)

TIPS FOR CREATING A BUSINESS MODEL CANVAS

- Start by focusing on user needs and value propositions
- Canvases can be created per-product in large organisations
- Use different colours to represent current and future states
- Use colours to show relationships
- Cross-functional activity with developers and stakeholders
- Not everything an organisation does or has is key
- Run practice workshops in your organisation

LEARN MORE

- **Business Model Generation** [book] - https://www.amazon.com/Business-Model-Generation-Visionaries-Challengers/dp/0470876417
- **Empathetic Software Development Guided by the Business Model Canvas** - http://ntcoding.co.uk/blog/2015/03/empathetic-software-development-guided
- **Strategyzer Business Model Canvas download** - https://strategyzer.com/canvas

IMPACT MAPPING

A common problem faced by many organisations is the mindset that managers create work and software engineers do the work. However, this results in incorrect assumptions on both sides. Managers make assumptions about how product features should be implemented, and engineers make incorrect assumptions when interpreting ambiguity.

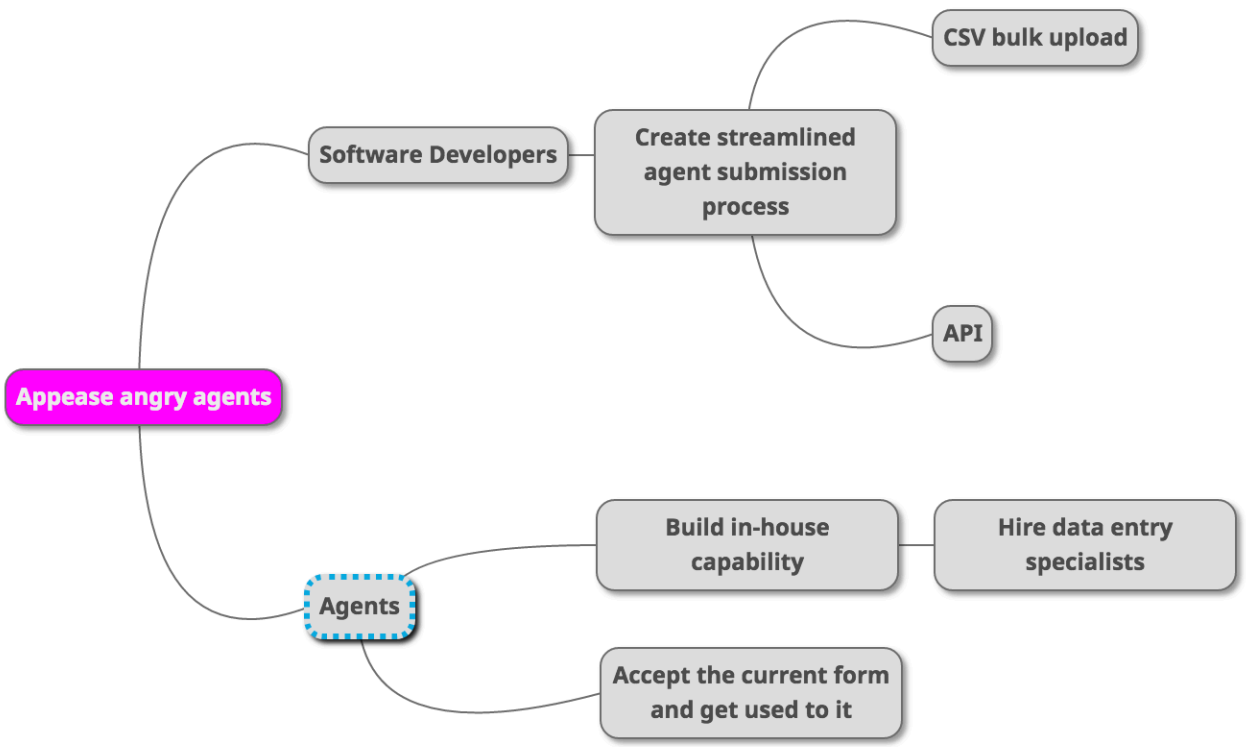
Impact Mapping solves these problems by making discovery highly collaborative. Engineers and managers rewind assumptions, start with the fundamental business problem, and then iterate on a range of potential solutions.

Impact maps are mind maps starting with a desired goal, then identifying potential actors, impacts and deliverables needed to achieve the goals.

DDD practitioners who practice impact mapping have a greater understanding of what is core to the business. They are better informed at deciding where to apply valuable effort exploring and modelling the domain.

AN EXAMPLE IMPACT MAP

The following impact map was created by an ecommerce organisation who were trying to find the best solution to resolve constant complaints they were receiving from angry agents. Note the flow from left to right: goal -> actors -> impacts -> deliverables.



TIPS FOR CREATING IMPACT MAPS

- Focus clearly on the goal before suggesting any impacts or deliverables
- Invite a cross-functional mix of people
- Run workshops in your organisation
- Think broadly how you can achieve the goal - sometimes the best solution may not be writing software
- Create as many ideas as possible first then narrow down options later

LEARN MORE

- **Impact Mapping** [book] - https://www.amazon.com/Impact-Mapping-Software-Products-Projects/dp/0955683645
- **Impact Mapping** [talk] - https://www.youtube.com/watch?v=X6gLICX7sl8
- **www.impactmapping.org/**
- **MindMup** - https://www.mindmup.com/

USER RESEARCH AND TESTING

With a strong focus on modelling domains, user needs are often overlooked in favour of fancy tactical patterns. But DDD practitioners who aspire to create relevant models that deliver value, place a great emphasis on understanding the needs of customers and how the domain provides a benefit to them.

One approach to better understanding user needs is to make user research and testing a whole team activity. Software developers should attend user research sessions or watch videos of user research and testing sessions in which customers talk or interact with software systems they have built.

USER RESEARCH AND TESTING TIPS

- Schedule a 1 hour meeting every iteration for all team members to watch user research and testing videos together
- Evangelise the benefits of user research to your peers to drive adoption in our organisation
- Understand the intricacies of the user researcher's job
- Create personas

LEARN MORE

- **How User Research Improves Service Design** - https://www.gov.uk/service-manual/user-research/how-user-research-improves-service-design
- **UK Government Digital Service User Research Blog** - https://userresearch.blog.gov.uk/
- **User Research at CoopDigital** - https://digitalblog.coop.co.uk/2016/07/25/user-research-at-coopdigital/amp/
- **Sense and Respond** [book] - https://www.amazon.com/Sense-Respond-Successful-Organizations-Continuously/dp/1633691888
- **Lean UX** [book] - https://www.amazon.com/Lean-UX-Designing-Great-Products/dp/1491953608

OTHER TECHNIQUES

- **The Product Strategy Canvas** - http://melissaperri.com/2016/07/14/what-is-good-product-strategy/
- **The Value Proposition Canvas** - https://strategyzer.com/canvas/value-proposition-canvas
- **The Lean Canvas** - https://leanstack.com/leancanvas
- **Wardley Value Chain Mapping** - http://blog.gardeviance.org/2015/02/an-introduction-to-wardley-value-chain.html

ADDITIONAL RESOURCES

- **Canvanizer** - https://canvanizer.com/
- **Value Proposition Design** [book] - https://www.amazon.com/Value-Proposition-Design-Customers-Strategyzer/dp/1118968050/
- **The Lean Startup** [book] - https://www.amazon.com/Lean-Startup-Entrepreneurs-Continuous-Innovation/dp/0307887898
- **Crossing the River by Feeling the Stones** [talk] – https://vimeo.com/189993565

MODELING CORE DOMAINS

COLLABORATIVE PRACTICES FOR UNDERSTANDING AND EXPLORING HIGH VALUE AREAS OF THE PROBLEM DOMAIN

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN

DDD FLAVOURED BDD

Behaviour-Driven Development (BDD) is an outside-in approach to creating a shared vision between business, product, and technical experts by talking through the different use cases to be implemented in software. BDD enables DDD practitioners and domain experts to have richer conversations that lead to greater domain insights.

When combined with DDD by using the ubiquitous language to describe scenarios, BDD helps to reinforce the conceptual DDD model leading to greater chances of better strategic and tactical designs.

Mastering three constructs will help you to maximise the value you get out of BDD. User stories frame the customer need, acceptance criteria define business rules, and scenarios provide examples of inputs and outputs to demonstrate the validity of business rules. Scenarios are typically expressed using Given, When, Then syntax, but this is not mandatory.

BDD does lack a focus on success criteria so you may also want to consider supplementing with Hypothesis-Driven Development.

AN EXAMPLE OF DDD FLAVOURED BDD

The following is an example of using DDD
flavoured BDD in the exploration of a new government tax system.

Note how words in italics are domain terms taken from the business glossary.

User Story

Viewing *Detailed Valuation*
As a *ratepayer*
I want to see a *detailed valuation* for my properties
So that I know if my *business rates bill* is too high

Acceptance Criteria

Verification Level 2 Permits Access to *Detailed Valuation*

Scenario

Given I have verified my *personal identity* to *level 2*
And I have *claimed* my property
When I request the *detailed valuation*
Then I should see the *rateable value*
And I should see all *attributes* of all *hereditaments*

TIPS FOR CREATING BDD SCENARIOS

- Start with use cases that provide most value to customers. Align with Business Model Canvas, product strategy, mission statement, etc.
- Focus on customer needs and business benefits not system internals
- Obsess over using precise phrases and domain terminology accurately
- Invite a cross-functional audience containing at-least domain, product, testing, and technical experts

LEARN MORE

- **What is in a Story?** - https://dannorth.net/whats-in-a-story/
- **Behaviour Driven Development** - https://lizkeogh.com/behaviour-driven-development/
- **BDD by Example** [talk] - https://www.youtube.com/watch?v=e2VfPOSLd9k
- **Specification by Example** [book] - https://www.amazon.com/Specification-Example-Successful-Deliver-Software/dp/1617290084
- **50 Quick Ideas to Improve Your User Stories** [book] - https://www.amazon.com/Fifty-Quick-Ideas-Improve-Stories/dp/0993088104

DOMAIN USE CASE DIAGRAMS

One of the best ways for developers to comprehend problem domains is to collaboratively model use cases with domain experts. Domain use case diagrams are an informal and flexible tool for working with domain experts to visualise use cases.

These diagrams are useful for exploring the internals of a use case, so follow on nicely from BDD scenarios that focus on external behaviours.

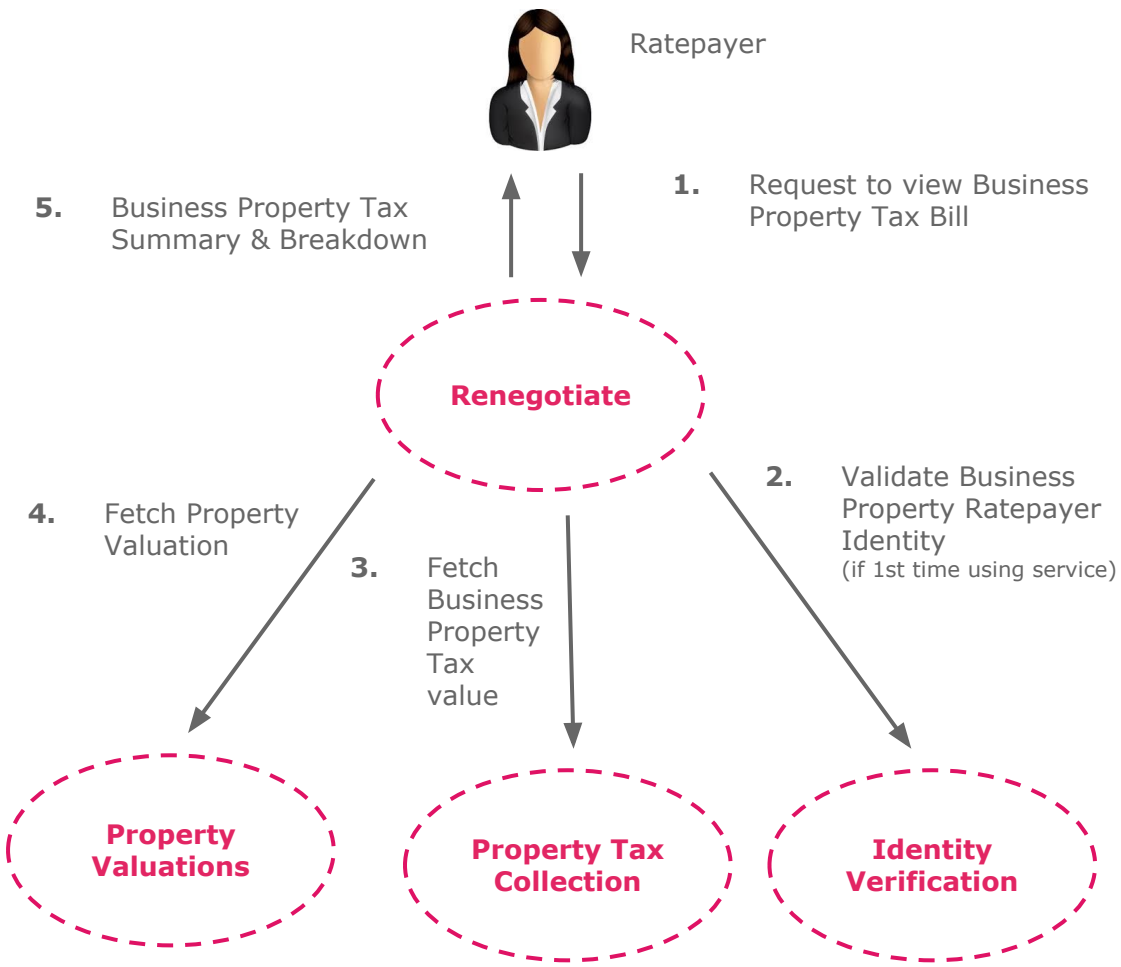
DDD practitioners rely on domain use case diagrams as an important step in discovering subdomains. By visualising the different processes and behaviours in a domain, DDD practitioners can see things that are related and belong to the same subdomain. Use case diagrams can be used to explore a domain to find subdomains, or after subdomains have been identified.

Domain use case diagrams also provide the blueprint of system architecture. For example, the interactions between subdomains on the diagram become events flowing through the software implementation.

You can instead consider UML use case diagrams if you prefer formal notation.

AN EXAMPLE DOMAIN USE CASE DIAGRAM

This domain use case diagram represents the core use case of a ratepayer seeking to view their tax bill produced by a government department.



TIPS FOR CREATING DOMAIN USE CASE DIAGRAMS

- Create domain use case diagrams collaboratively with domain experts
- There is no formal notation
- It is ideal but not mandatory to show subdomains or contexts
- Create use case diagrams in combination with BDD scenarios
- Don't clutter use case diagrams with technical details. Create multiple diagrams instead.
- Use ubiquitous language

LEARN MORE

- **Domain-Driven Architecture Diagrams** - http://ntcoding.co.uk/blog/2015/08/domain-driven-architecture-diagrams
- **Basics of UML Use Case Diagrams** - https://creately.com/diagram-type/article/basics-uml-usecase-diagrams
- **Patterns, Principles, and Practices of Domain-Driven Design** [book] - https://www.amazon.com/Patterns-Principles-Practices-Domain-Driven-Design/dp/1118714709/

BIG PICTURE EVENT STORMING

Big picture event storming is a highly-collaborative, highly-interactive way for a cross-functional audience to explore a problem domain by creating a timeline-like flow of post-it notes representing key events in the domain.

Event storming can lead to a wide variety of insights across all parts of an organisation or system because attendees will learn about less familiar parts of the domain. Different teams may realise their assumptions about other parts of the system were incorrect leading to organisational reshuffles, new technical models, etc.

The defining characteristic of event storming is the huge number of post-it notes and wall space required. Event storming uses different coloured post-its to represent modelling constructs including commands, events, and external systems.

TIPS FOR EVENT STORMING

- You need lots of wall space and post-it notes.
- Bring together a variety of people who understand different parts of the process/system
- Try not to enforce structured patterns too early otherwise people switch off
- Zoom in on areas where there are lots of debate and discussions
- Don't talk about technical patterns, like aggregates, with non-technical people

LEARN MORE

- **Introduction to Event Storming** - http://ziobrando.blogspot.co.uk/2013/11/introducing-event-storming.html
- **Event Storming** [book] - https://leanpub.com/introducing_eventstorming
- **Event Storming** [talk] - https://www.youtube.com/watch?v=veTVAN0oEkQ
- **Event Storming for Fun and Profit** [talk] - https://skillsmatter.com/skillscasts/8003-event-storming-for-fun-and-profit

CORE DOMAINS DIAGRAM

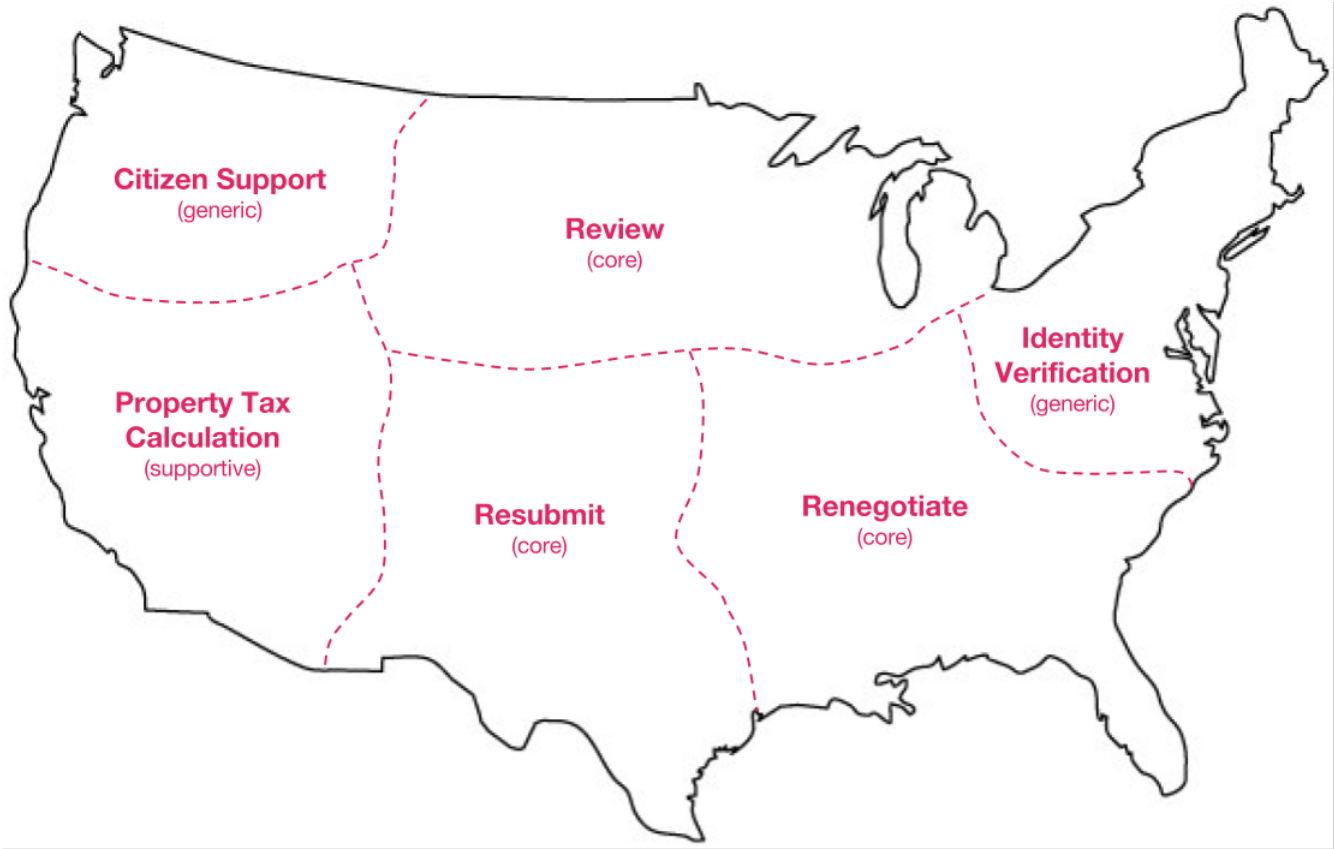
Having a working hypothesis of the subdomains believed to exist in the domain, and which of them are core, helps DDD practitioners apply their modelling and engineering efforts where there is greatest return on investment.

A core domains diagram is a simple, informal diagram showing each subdomain and its importance, with an emphasis on accentuating core domains. The importance of this visualisation is that it represents an agreement between domain, business, and technical experts of what the most important areas of the problem domain are.

In general, subdomains are core if they are key to business success, supportive if they are necessary but provide no competitive advantage, and generic if they can be bought off the shelf or require little innovation. However, in reality, many subdomains do not fit neatly into each category so it's best to treat core, supportive, and generic as guidelines as opposed to hard rules. Importantly, what is core can change over time, too.

AN EXAMPLE CORE DOMAINS DIAGRAM

The following core domains diagram represents the subdomains that could exist in a government agency. It uses the shape of the USA to lay out subdomains akin to states, but you are free to use whatever shape you prefer.



TIPS FOR CREATING CORE DOMAINS DIAGRAMS

- Just create a quick, informal sketch. It doesn't have to be fancy
- Make it easy for others to understand where return on efforts are greatest
- Ensure you are using 100% ubiquitous language.
- Sometimes a subdomain will be core to the business but supportive for the current project and vice-versa. You can show both ratings if you like
- Work collaboratively with business and domain experts or at-least get their approval after creating the diagram

LEARN MORE

- **DDD: Strategic Design: Core, Supporting, and Generic Subdomains** - http://blog.jonathanoliver.com/ddd-strategic-design-core-supporting-and-generic-subdomains/
- **Domain-Driven Architecture Diagrams** - http://ntcoding.co.uk/blog/2015/08/domain-driven-architecture-diagrams

OTHER TECHNIQUES

- **Design Level Event Storming** - http://eventstorming.com/
- **Exploratory code prototyping with domain experts** - https://gojko.net/2009/03/12/qcon-london-2009-eric-evans-what-ive-learned-about-ddd-since-the-book/
- **Ad-hoc knowledge crunching** - https://leanpub.com/theanatomyofdomain-drivendesign

ADDITIONAL RESOURCES

- **Modelling Heuristics** [presentation] - http://verraes.net/2014/11/modelling-heuristics/
- **Domain-Driven Design case study** [video] - https://www.youtube.com/watch?v=7MaYeudL9yo
- **Towards Modelling Processes** [talk] - https://www.youtube.com/watch?v=raTX3S_LFI8

THE STRATEGIC PRACTICES OF DDD
BY NICK TUNE (@NTCODING)

ALIGNING ORGANISATION AND TECHNICAL BOUNDARIES

STRATEGIC PATTERNS AND PRACTICES FOR DESIGNING AUTONOMOUS TEAMS AND SERVICES

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN

BOUNDED & AUTONOMY CONTEXTS

Breaking down a large problem domain into multiple smaller models prevents unrelated concepts becoming tangled and causing confusion due to ambiguity. Traditionally, DDD has solved this problem with the use of bounded contexts aka linguistic boundaries.

Bounded contexts are arbitrary boundaries created in the solution space. The goal is for each bounded context to have a clear responsibility and an explicit model.

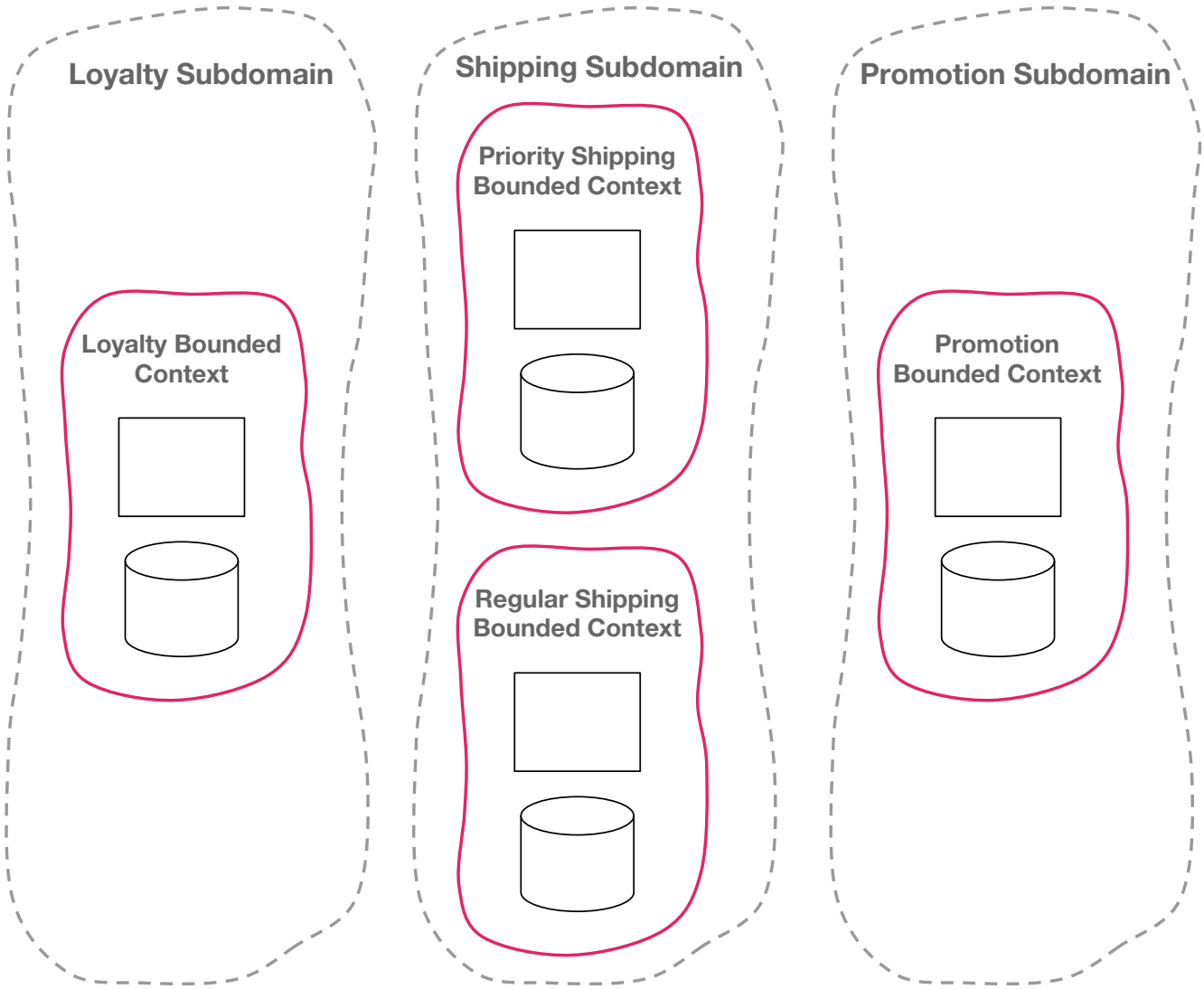
An explicit model is the key to minimising ambiguity and keeping high alignment between the spoken model, the conceptual model and the code model. If the same term or phrase in your domain has multiple definitions, it's a big hint that you need multiple bounded contexts. A common anti-pattern is the User God Object that accrues behaviours from multiple contexts.

Ideally, there is a 1:1 mapping between bounded contexts and subdomains to create high-alignment between the problem space and solution space, however this is not always feasible or realistic.

Historically, bounded contexts were modules within a monolithic codebase, but with the rise of SOA and Microservices, bounded contexts have recently become a popular heuristic for defining boundaries in distributed systems.

AN EXAMPLE OF BOUNDED CONTEXTS

This diagram shows a sample of the bounded contexts existing in an ecommerce domain. Note how a 1:1 mapping between subdomains and bounded contexts is not always desirable or feasible.



AUTONOMY CONTEXTS

Autonomy contexts extend bounded contexts to include organisation design because a poor organisation design will preclude the ability to create an effective technical design. Therefore, organisational and technical boundaries should coevolve.

Autonomy contexts are based on the belief that autonomy is the goal of boundaries. Autonomy contexts should encapsulate things that change together for business reasons. Accordingly, most work should occur within a single autonomy context, and be owned by a single team, rather than across autonomy contexts, requiring collaboration from multiple teams.

Autonomy contexts combine insights from DDD, Theory of Constraints, and Lean. You can start with bounded contexts, but as work flows through the organisation look for teams becoming bottlenecks. Then use DDD heuristics and ToC to redesign organisational and technical boundaries to remove the bottleneck.

In addition to language, many criteria can be used to guide the design of autonomy contexts, including:

- Business process steps
- Business capabilities
- Ownership, flow, and uniqueness of data
- Bottlenecks
- Business value of context
- Physical location of teams

TIPS FOR DESIGNING AUTONOMY CONTEXTS

- A key priority is delivering value to customers sustainably faster. Organisational and technical boundaries should serve this need above creating pretty models.
- Autonomy contexts may contain more than 1 service, database, etc.
- Minimise dependencies between teams so most of their time is spent on adding value rather than collaborating
- Organisational and technical boundaries should coevolve - code insights should affect team boundaries, and flow of work through the organisation should affect technical boundaries
- Autonomy contexts should be designed continuously and collaboratively by business, domain, and technical experts
- There is no perfect design – experiment with multiple models

LEARN MORE

- **Finding Service Boundaries: The One Rule That Matters** - http://ntcoding.co.uk/blog/2017/01/finding-service-boundaries-one-rule
- **Domain-Driven Design: Bounded Contexts, Modelling** - http://verraes.net/2014/02/domain-driven-design-basics/
- **Finding Service Boundaries - Illustrated in Healthcare** [talk] - https://vimeo.com/113515335

CONTEXT MAPS

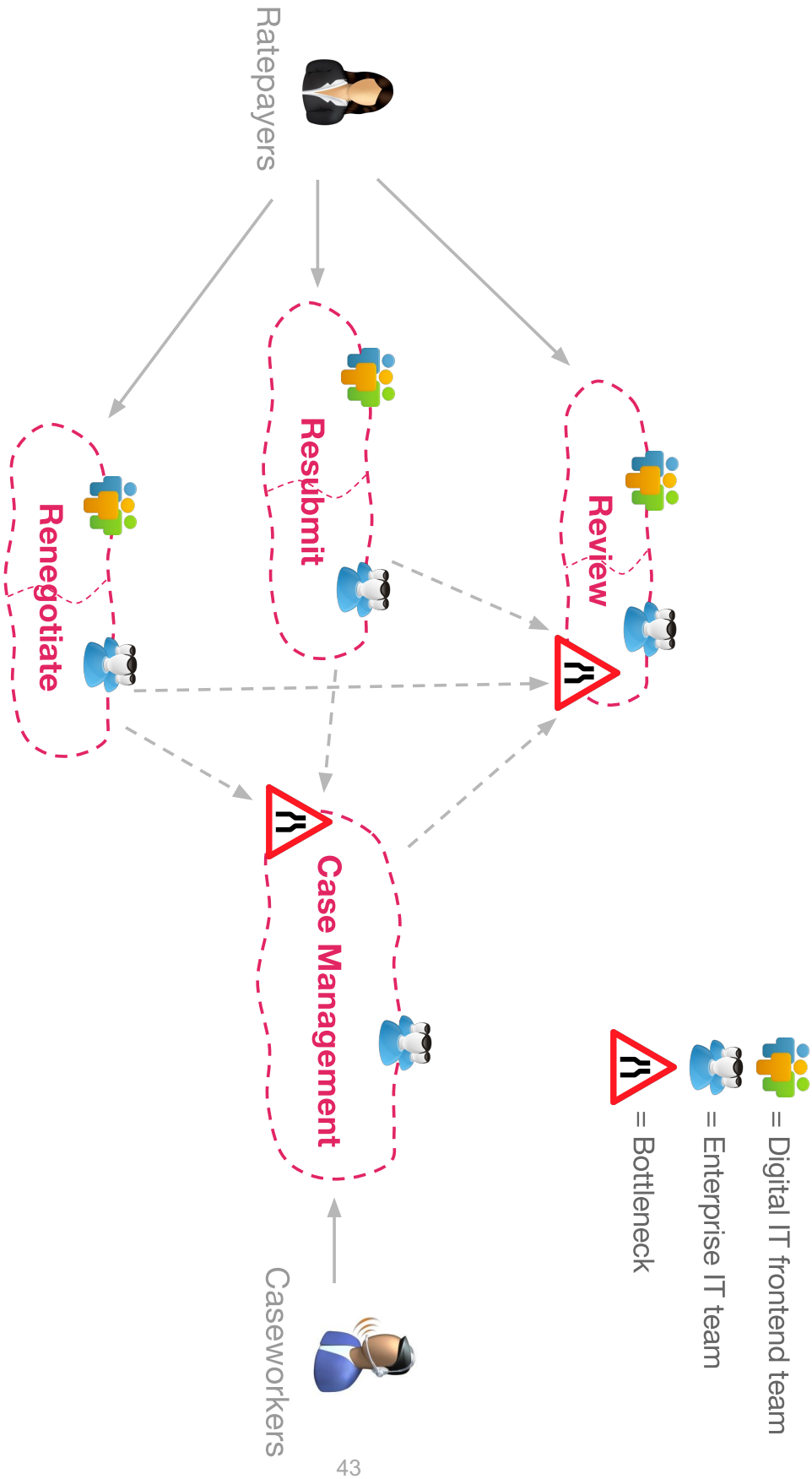
Context maps are a visualisation of your high-level strategy for solving the business problem. They show the organisational and technical boundaries you believe will lead to the most effective solution for sustainably delivering value.

Good context maps create alignment and trust between business, domain, and technical experts and provide the basis for strategic conversations.

There is no official notation for context maps, nor rules that govern them. You can visualise any information that highlights the relationship between domain, team, and technical boundaries.

AN EXAMPLE CONTEXT MAP

The following context map visualises the logical boundaries in the solution space, emphasising the organisational alignment and the bottlenecks. However, context maps can also show a variety of other information including technical integration strategies, external dependencies, and reasons for integration.


Ratepayers
Renegotiate
Resubmit
Review
Case Management
Caseworkers
= Bottleneck
= Enterprise IT team
= Digital IT frontend team

TIPS FOR CREATING CONTEXT MAPS

- Start with too much information and remove low value information later
- Create separate diagrams for current state and future state
- Vital to use ubiquitous language
- Get feedback from business, domain, and technical experts who are covered or affected by areas of the map
- Don't get too distracted by creating perfect boundaries at first
- Show organisational factors - the team owning the context, relevant stakeholders, location of team etc.

LEARN MORE

- **Context Mapping: Life Expectancy** - http://verraes.net/2015/04/context-mapping-life-expectancy/
- **Bandwidth and Context Mapping** - http://verraes.net/2014/01/bandwidth-and-context-mapping/
- **Patterns, Principles, and Practices of DDD** [book] - https://www.amazon.com/Patterns-Principles-Practices-Domain-Driven-Design/dp/1118714709/

VALUE STREAM MAPS

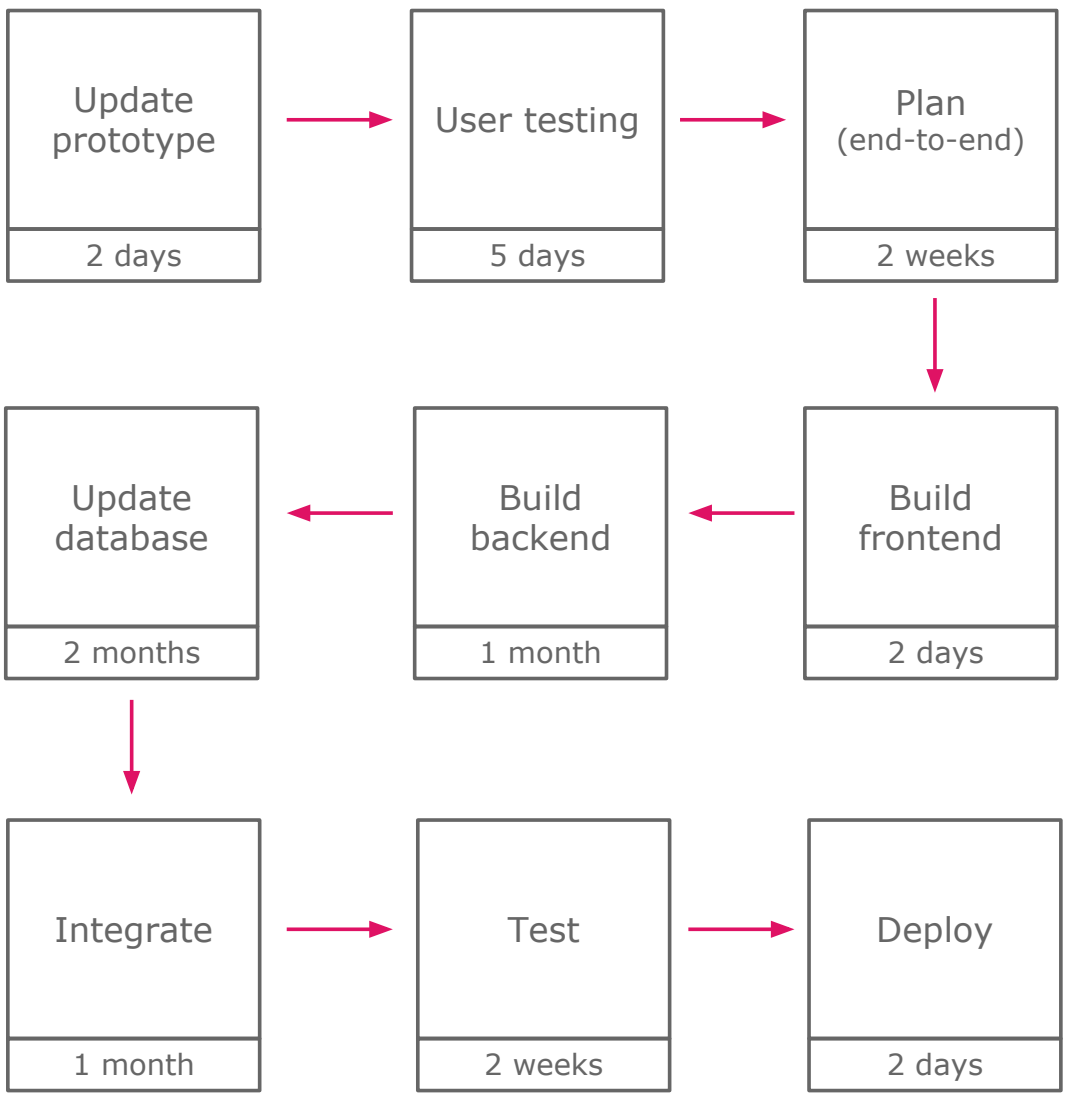
Value stream mapping provides a Systems Thinking approach to software development. By mapping out the stages in a process and calculating how much time is spent on each, insights can be gained about the efficiency of the entire process. Instead of each process step optimising for efficiency, the entire flow can be optimised for efficiency.

DDD practitioners benefit hugely from value stream maps because value stream maps provide feedback about strategic design choices. If a specific stage in the value stream is slow - acting as a bottleneck in the organisation - it can indicate the team and technical boundaries are suboptimal.

While language, bounded contexts, and modeling are key heuristics in Domain-Driven Design - pretty models are not enough. The most important criteria for an organisation is delivering positive business outcomes as frequently as possible. Combining DDD with Theory of Constraints and value stream maps leads to better outcomes and shorter lead times by encouraging Systems Thinking.

AN EXAMPLE VALUE STREAM MAP

This value stream map shows the value stream for a digital product requiring the input of three teams. This map accentuates the bottlenecks - the handovers and integration costs of the frontend, backend and database teams.



TIPS FOR CREATING VALUE STREAM MAPS

- Start with more information than you need and narrow the scope later
- Consider showing average times or time ranges for each stage
- Consider showing the amount of rework from each stage (often as a percentage)
- Consider showing wait times (time work spends waiting between each phase)
- Work collaboratively across functions to gain a range of insights

LEARN MORE

- **What is Value Stream Mapping?** - https://leankit.com/learn/kanban/what-is-value-stream-mapping/
- **Creating A Value Stream Map** - http://leanmanufacturingtools.org/551/creating-a-value-stream-map/
- **Value Stream Mapping** [book] - https://www.amazon.com/Value-Stream-Mapping-Organizational-Transformation/dp/0071828915

OTHER TECHNIQUES

- **DevOps Topologies** - http://web.devopstopologies.com/
- **Inverse Conway Maneuver** - https://www.thoughtworks.com/radar/techniques/inverse-conway-maneuver

ADDITIONAL RESOURCES

- **Alignment at Scale** [talk] - https://www.infoq.com/fr/presentations/lkfr-henrik-kniberg-keynote
- **Lean Enterprise** [book] - https://www.amazon.com/Lean-Enterprise-Performance-Organizations-Innovate/dp/1449368425
- **The Goal** [book] - https://www.amazon.com/Goal-Process-Ongoing-Improvement/dp/0884271951
- **Thinking in Systems** [book] - https://www.amazon.com/Thinking-Systems-Donella-H-Meadows/dp/1603580557
- **Building Microservices** [book] - https://www.amazon.com/Building-Microservices-Designing-Fine-Grained-Systems/dp/1491950358

DOMAIN-DRIVEN TECHNICAL STRATEGY

PRACTICES FOR DESIGNING AND COMMUNICATING A TECHNICAL STRATEGY ALIGNED WITH THE BUSINESS MISSION

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN

SYSTEM CONTEXT DIAGRAM

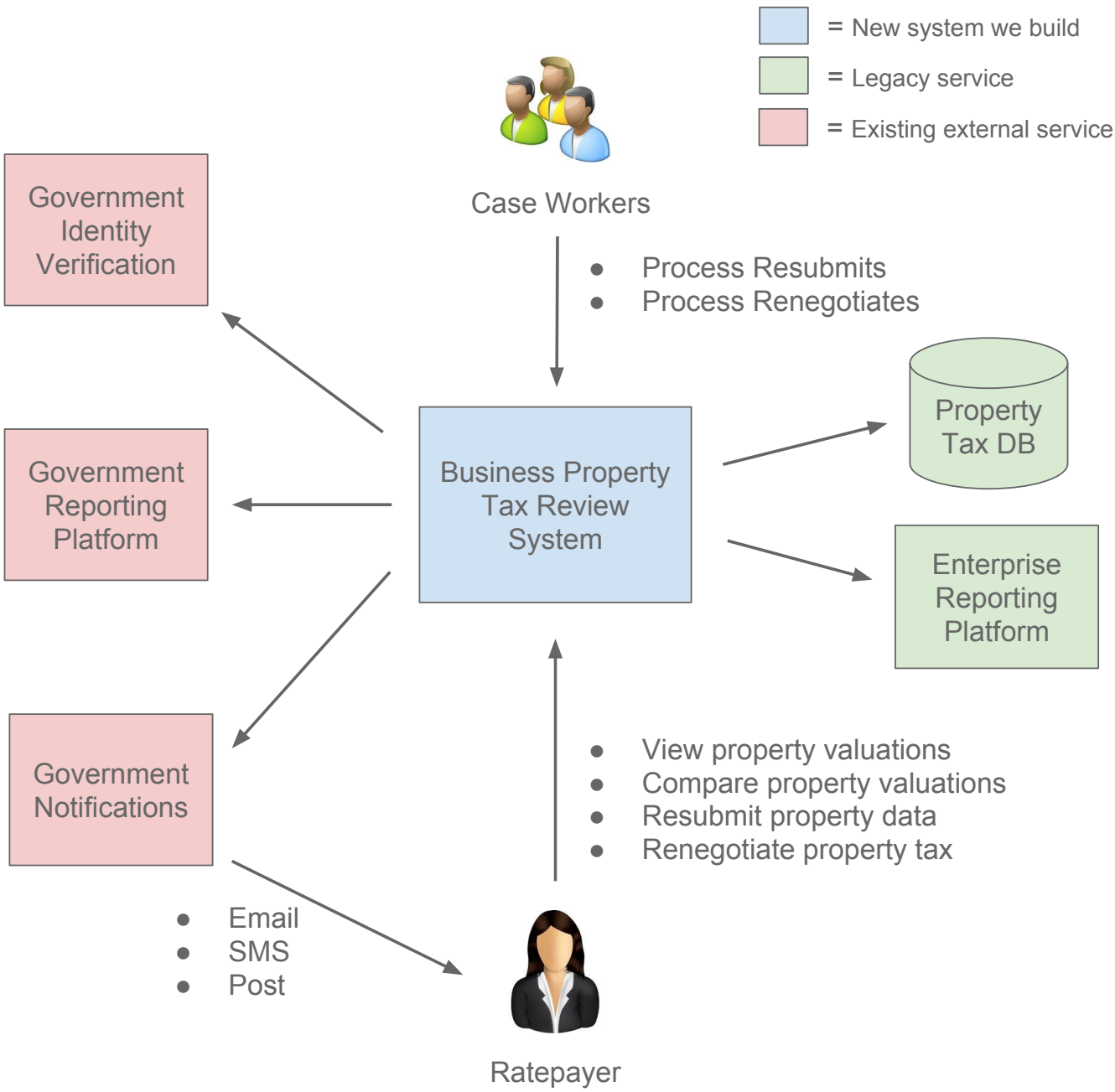
Providing a high-level vision of your technical strategy can help all technical and non-technical peers understand how your architecture aligns with the business vision. Importantly, your high level technical vision can communicate technical and organisational risks that are not inherent in the problem domain and may not otherwise be understood by business and domain experts.

A system context diagram is the highest level diagram of your architecture. In the middle of the diagram is the entire system you are proposing to build. Additionally, the diagram shows the users of the system, the use cases they care about, as well as external systems which must be integrated with.

The system context diagram is not a traditional DDD diagram, however DDD practitioners find it a valuable tool for communicating challenges and risks that may otherwise fall through the cracks. The system context diagram is still a representation of your model, so the usual best practices of aligning with the business vision and using ubiquitous language still apply.

AN EXAMPLE SYSTEM CONTEXT DIAGRAM

The following system context diagram illustrates the high level architecture that could exist in a government tax department.



TIPS FOR CREATING SYSTEM CONTEXT DIAGRAMS

- Focus very high level on users and major dependencies
- Use ubiquitous language
- Explain major use cases
- Get feedback from business and domain experts
- Use colour codes and a key
- Don’t clutter with too much information
- Focus on creating a shared vision

LEARN MORE

- **Software Architecture for Developers** [book] - https://leanpub.com/software-architecture-for-developers
- **Visualise, Document & Explore your Software Architecture** [talk] - https://www.youtube.com/watch?v=0o9_zjZeJuE
- **Domain-Driven Architecture Diagrams** - http://ntcoding.co.uk/blog/2015/08/domain-driven-architecture-diagrams

AUTONOMY CONTEXT DIAGRAM

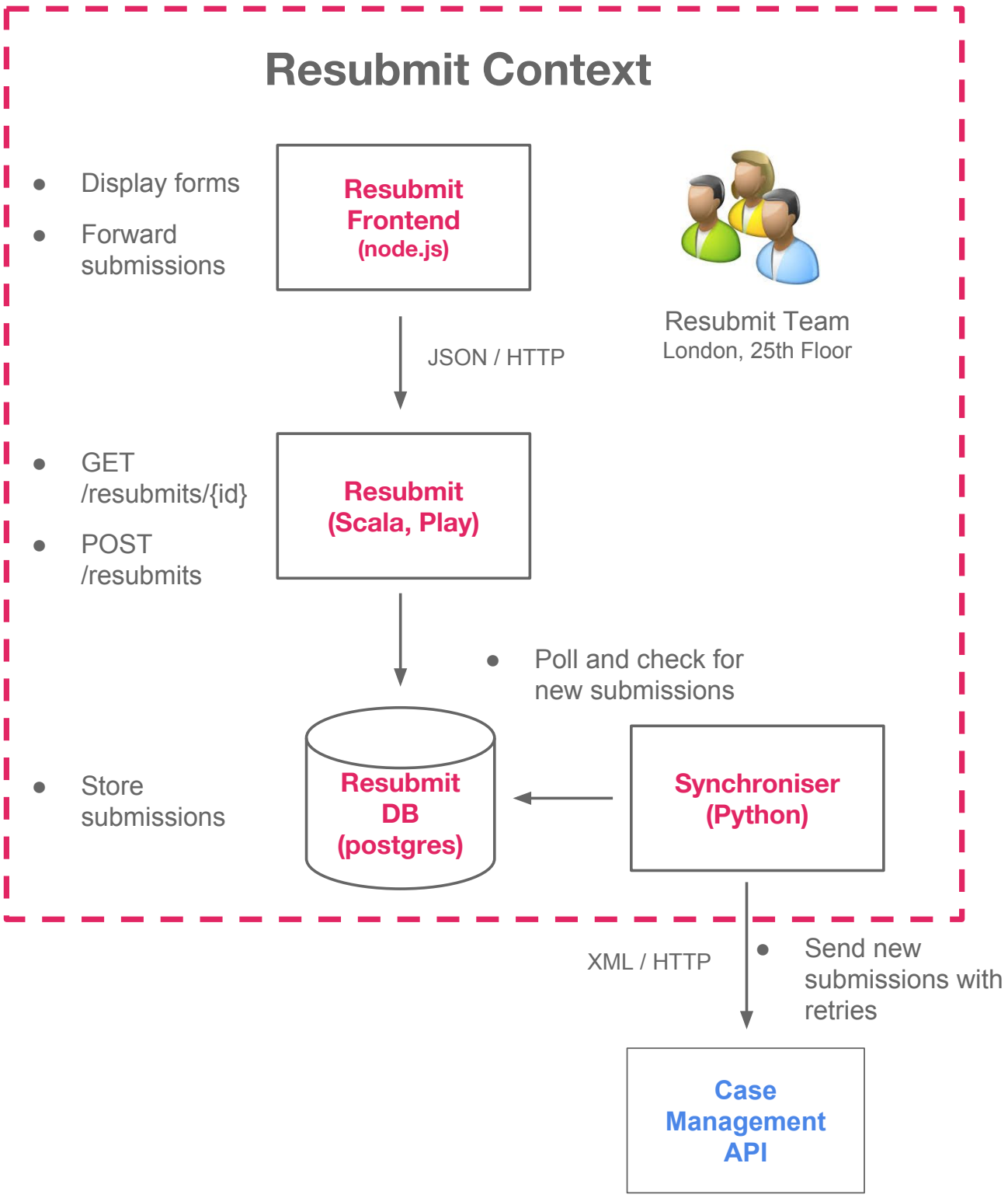
Autonomy context diagrams show the high-level organisational and technical building blocks that make up the context i.e. the team that owns the context and each technical service that the team owns.

The goal of autonomy context diagrams is to provide an at-a-glance view of the major architectural choices and responsibilities of the context. The diagram is intended to create a shared vision between the team that owns the context (or will be building it), but the diagram is also a good communication tool to share with other teams and for onboarding newcomers.

It is good practice to show technical detail on autonomy context diagrams, but it is important to also show alignment with the domain by naming services using domain terminology and explaining trade-offs.

AN EXAMPLE AUTONOMY CONTEXT DIAGRAM

The following autonomy context diagram shows key technical and organisational details for a single context in a government tax system.


Resubmit Context
Display forms
Forward submissions
Resubmit Frontend (node.js)
Resubmit Team
London, 25th Floor
JSON / HTTP
GET /resubmits/{id}
POST /resubmits
Resubmit (Scala, Play)
Poll and check for new submissions
Store submissions
Resubmit DB (postgres)
Synchroniser (Python)
XML / HTTP
Send new submissions with retries
Case Management API

TIPS FOR CREATING AUTONOMY CONTEXT DIAGRAMS

- Show every microservice/app owned by the context
- Communicate major technology choices which provide a useful high level overview to engineers in yours and other teams
- Communicate major organisational choices - team members, key stakeholders, physical location, etc.
- Add justifications to your diagram explaining the rationale for key technical and organisational choices
- It's ok to show touch points with key external services as long as the boundary of ownership is clear
- Domain and business experts aren't the primary audience, but it can still be useful to get their feedback
- Don't clutter the diagram with details of multiple different use cases, use technical use case diagrams instead

LEARN MORE

- **Domain-Driven Architecture Diagrams** - http://ntcoding.co.uk/blog/2015/08/domain-driven-architecture-diagrams
- **Software Architecture for Developers** - https://leanpub.com/software-architecture-for-developers

TECHNICAL USE CASE DIAGRAM

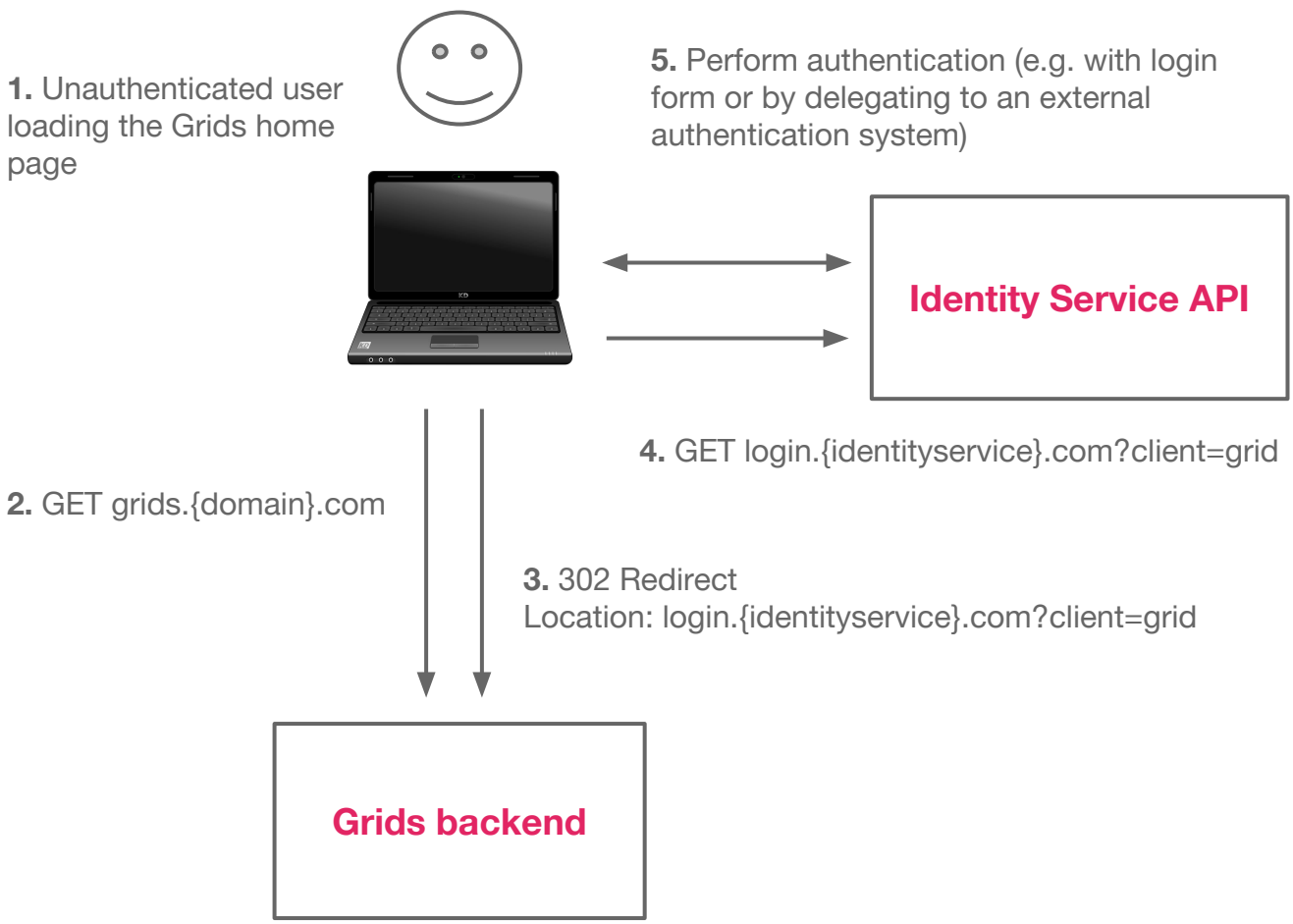
Sometimes it is important to show technical details alongside domain processes. For example, you have a legacy system making it difficult to implement a specific business use case, or you want to show how specific technologies or algorithms provide the capability for a key value proposition. In such scenarios, you can create technical use case diagrams.

Technical use case diagrams can be used at varying levels of granularity. They can show the flow of events between bounded contexts or they can zoom into a specific bounded context and show the flow of logic between modules, or even at the class level if necessary.

There are no official rules or notation for technical use case diagrams. Even though providing technical details, it's still important to use domain vocabulary and try to explain decisions on your diagram according to the business vision.

AN EXAMPLE TECHNICAL USE CASE DIAGRAM

The following technical use case diagram illustrates the authentication process for an analytics system. In addition to process information, it includes technical details including URLs, HTTP status codes, and technical components.



TIPS FOR CREATING TECHNICAL USE CASE DIAGRAMS

- Focus on a specific use case
- Show technical details, but align with the domain process
- Use ubiquitous language
- Use colour schemes and keys to improve communication
- Be careful of adding too much detail. 7 boxes is normally a sign you might be showing too much information
- There is no formal notation, but you can take influence from UML use case diagrams

LEARN MORE

- **Domain-Driven Architecture Diagrams** - http://ntcoding.co.uk/blog/2015/08/domain-driven-architecture-diagrams.html
- **UML 2 Use Case Diagrams** - http://www.agilemodeling.com/artifacts/useCaseDiagram.htm

OTHER TECHNIQUES

- **UML sequence diagrams** - http://www.agilemodeling.com/artifacts/sequenceDiagram.htm
- **Enterprise Architecture** - https://www.forrester.com/Enterprise-Architecture

ADDITIONAL RESOURCES

- **Structurizr** - https://www.structurizr.com/
- **The Need for Speed: Enabling DevOps through Enterprise Architecture** [presentation] - https://www.slideshare.net/willevans/the-need-for-speed-enabling-devops-through-enterprise-architecture
- **Coding the Architecture** - http://www.codingthearchitecture.com/authors/sbrown/
- **NTCoding Architecture Blog** - http://ntcoding.co.uk/blog/labels/Architecture
- **Visualise, Document & Explore your Software Architecture** - https://www.youtube.com/watch?v=0o9_zjZeJuE

@NTCODING
NTCODING.CO.UK/WORKSHOPS
#DDDESIGN