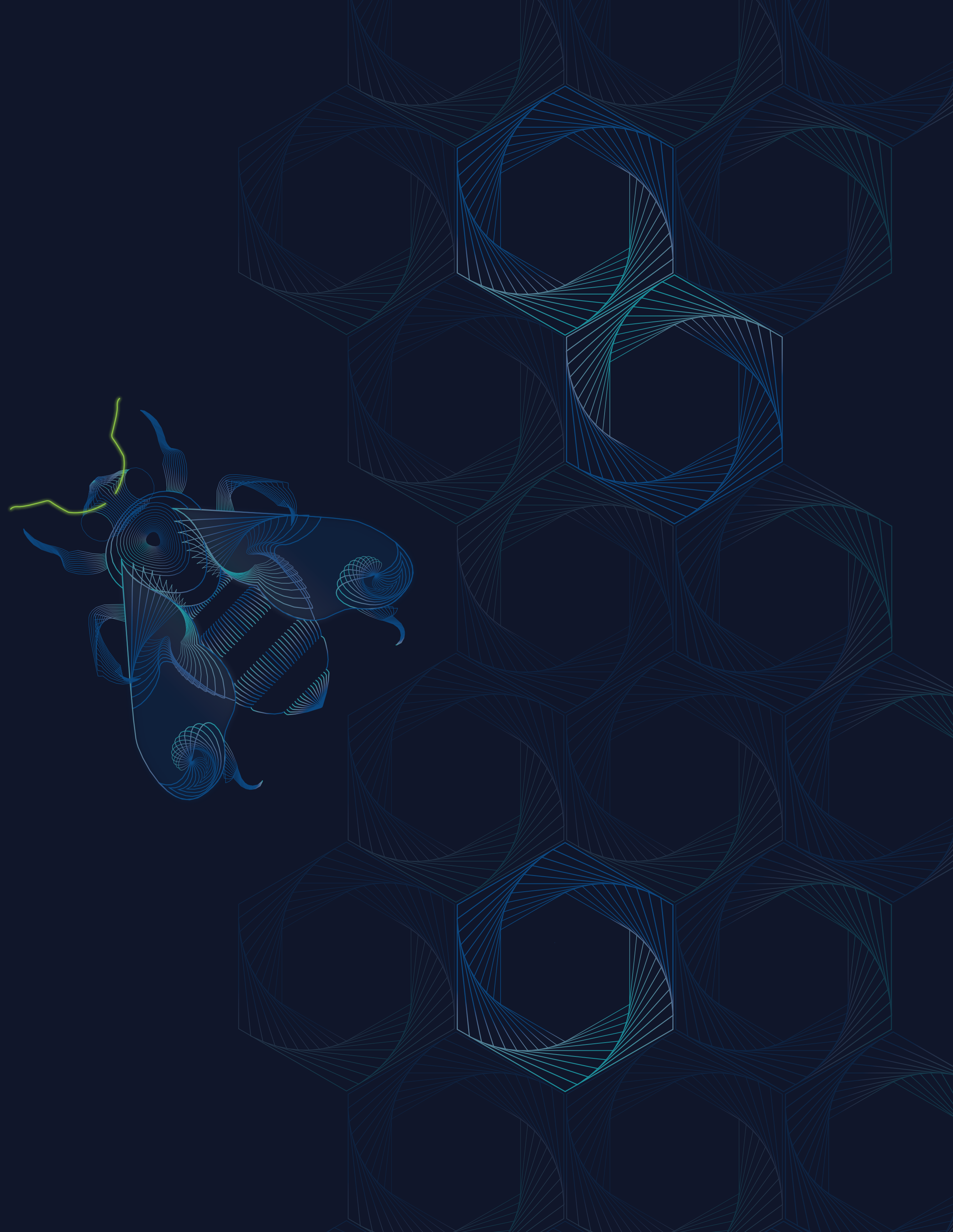
entelect

Elements of technology project success

Practices which are key to meeting objectives and achieving a return on investment.



possibilities

Over the years we've seen many technology projects fail to deliver a return. In some cases, products don't actually go on to solve the problems they were originally commissioned for, or are shelved before completion. Other times they reach production only to suffer from a lack of adoption. Why?

Sharing Entelect's experience in industry, these are **our own lessons for the major factors which influence project success** through the lenses of user adoption, time and budget. We explore the practices, ways of work and mindsets which are consistent across successful teams, and highlight some of the critical decisions involved in each project we undertake.

Build the right product

Will the proposed solution actually go on to solve the problem?

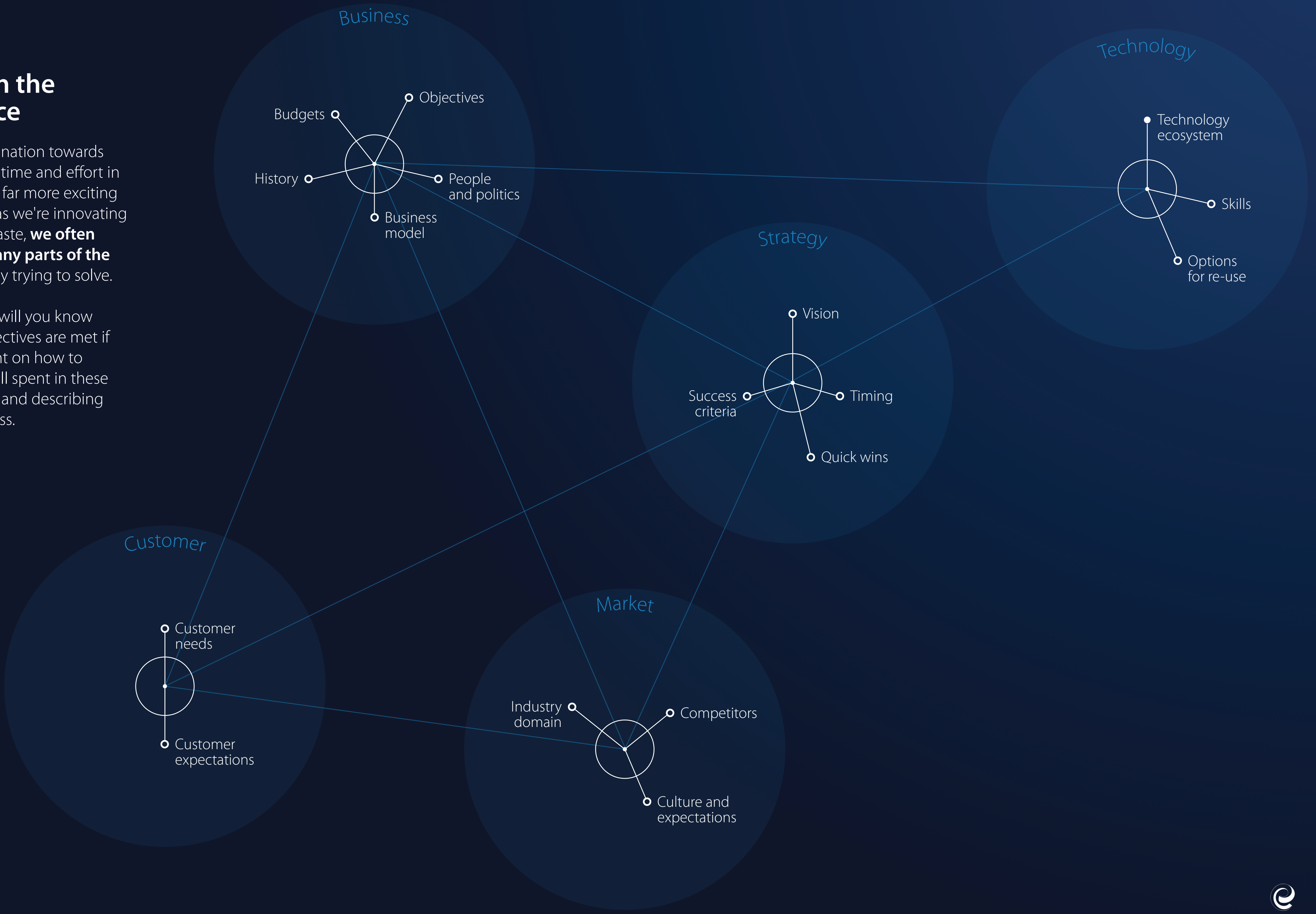
A broad analysis is required for clear understanding of the customer perspectives, the business objectives, and technology constraints or opportunities.

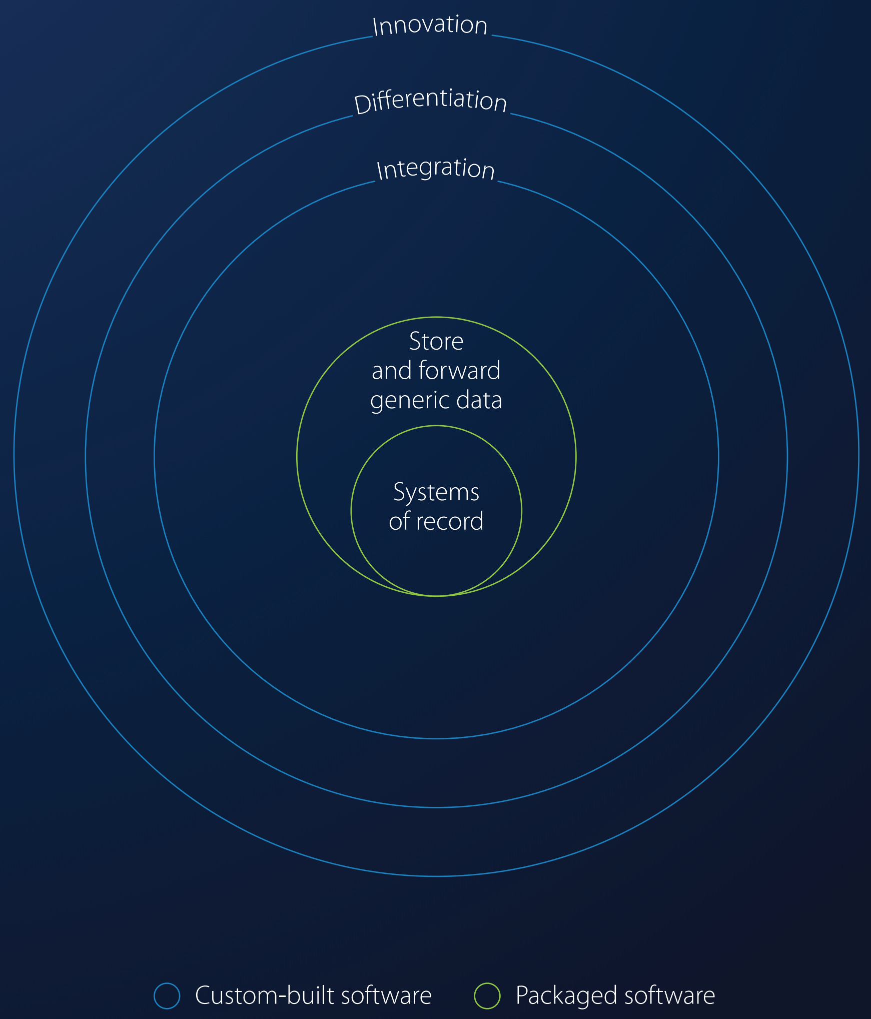
Depth on every topic is not as important as good coverage, as the intention is to discover roadblocks early, not to exhaustively specify.

Spend time in the problem space

We have a natural inclination towards spending more of our time and effort in the solution space. It's far more exciting and interesting there as we're innovating and building. In this haste, **we often leave unexplored many parts of the problem** we're actually trying to solve.

Define success. How will you know whether business objectives are met if you can't agree upfront on how to measure it? Time is well spent in these early phases debating and describing clear metrics for success.





Build vs buy

Deciding whether to build your own bespoke technology, or licensing and installing something that already exists goes beyond the comparisons of total cost of ownership. **There are strategic questions to be asked** of IP ownership and control, competitive differentiation, and the longer-term effects on an enterprise architecture.

When you're buying
You're ready to adopt something that exists, either because it's cheaper or because it can solve the problem faster. **You're not reinventing the way something has always worked**, and are comfortable to change your business processes to suit the product.

When you're building
You have original ideas for the technology that runs your business, or through which your customers interface with you. You're forging new ground, can't or won't change your processes and need a solution built around your needs. **You are prepared to spend the time and resources to build and own a bespoke software asset**.

> **"If you are truly innovating, you don't have a prototype you can refer to."**
> - Jony Ive, Chief Design Officer of Apple Inc.

A balanced portfolio: Core-satellite approach
Similar to the investment world, where a portfolio is carefully designed to balance risk and return, so too should we be **finding the optimum balance in our technology strategies**.

At the 'core' of the strategy are assets that are cost-effective, proven, robust and widely used. This is usually where you **need dependable and functional packaged software as the building blocks of your architecture**.

At the 'satellites', you need your own ideas and agency, **custom solutions which allow for innovation and differentiation**. These are unique and memorable experiences which offer greater impact and reward.

A customer seat at the table

Naturally, our businesses focus on things like innovation and differentiation of products or services that can gain market attention. However, interesting or novel ideas don't imply actual consumption. This means we need to more frequently test and adapt our ideas between the concept and the product reality to have any real confidence that there will be longer term traction.

We must create a customer seat at our table. We need to continually engage them throughout the project, to **understand their perspective, and frequently validate our evolving product**.

Be aware of, and try to avoid "expert arrogance", where assumptions are made about what customers want, informed by experience in an industry or domain. It is always worth interrogating the assumption.

Customer-focus is lip-service unless you're able to invite continuous feedback and validation from real users. Including experience design as a component of your project will **invite the customer voice into the team**.

> **"Obsessing over customer experience is the only long-term defensible competitive advantage"**
> - Jeff Bezos, Founder of Amazon.com

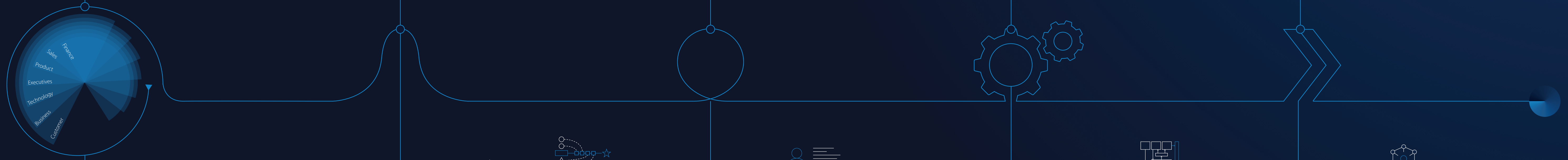
Pragmatic analysis

An effective analysis phase requires us to cover a lot of ground.

Within a timeframe which is deliberately limited to avoid analysis paralysis, the goal is to achieve a level of confidence across a mix of stakeholders that the problem is well understood. We need to agree on a shared vision, and uncover any major obstacles early. These are the tools and artefacts we recommend prioritising during an analysis window.

If you need help with preparation and analysis for a major project, reach out to us about Entelect's Design Sprints

› solutions@entelect.co.za



Aligning stakeholders

Strategy and tactics | *Workshop*

Executive-led discussions around purpose and business objectives to crystallise thinking and draw out initial tactical ideas.

Organisational Chart | *Workshop*

An org map plots all the people, stakeholders and roles within the business.

Shared understanding

Business map | *Workshop*

Create an overall map of the specific business context, including goals, human and system actors and general processes.

Design Sprints

Leverage a diversity of perspectives through a structured but rapid series of workshops and activities to problem-solve and validate solutions together.

Test and validate

Personas

Define distilled profiles for the different user roles and potential variety in them.

User Journeys

Create a user journey map for each meaningful persona defined, including system touch points and business rules.

Prototypes

Sketch low-fidelity, or high-fidelity mockups and stitch them together as an interactive prototype to solicit feedback from a more tangible product at low effort and cost.

Technical feasibility

Logical Architecture

Illustrate the general logical and technical composition of components in the software solution.

Data Models

Construct a diagram and a list of data inputs and outputs to build a concise well-understood foundation for handling data.

Process Models

Develop flow diagrams of business and system processes that are critical to the solution.

Way forward

Ways of work

Every project and greater technology team must have a consistent, reliable framework in place for how work flows between people and teams.

Flight Plan

Create a high-level flight plan that structures milestone deliverables over the course of the project.

Agile execution and budgeting

Agility promises efficiencies in the route taken.

Our aim is to minimise waste in the process of evaluating, adapting and delivering on an idea. These are smart investments as they can course-correct with customer feedback as the guide.

To enable this, executive-level thinking needs to embrace the MVP, and IT budgeting has to adapt.

MVP

Build the smallest thing that delivers customer value

Avoid over-investing by adopting the minimum viable product (MVP) approach, for both build and buy. This means delivering the smallest possible, but still meaningful, part of a product into the market to learn from real consumption metrics.

Critically, this doesn't mean "quick and dirty". The philosophy is **small on feature-set, not small on quality**. We want to avoid wasting money and effort on ideas that don't work.

We still need big thinking for the longer term, tempered by smaller goals in the shorter term that offer a coherent foundation for an incremental release roadmap ahead. **This mindset has to be shared by the management and development team**.

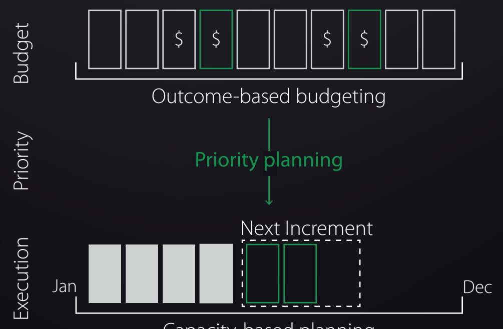
Rapid prototyping tools have come a long way in recent years. Ideas for systems and apps can be presented very convincingly, and demonstrated to a target audience for early validation of features and ideas prior to any real heavy implementation.

Agile execution versus the traditional IT budget

To support this mode of operation, tightly controlled annual budgets won't cope. This means **planning needs to be capacity-based, and fluid**. Each team (business and tech) must be empowered with outcome-based funding, and monitored iteratively.

This doesn't mean endless budgets, but rather more diligence on a consistent basis as we transition from on-time in-budget metrics for technology projects to **outcome and customer-focused funding which emphasises value**.

Financial feasibility must still be tested, and can be checked against lower precision estimates. For example you can usually calculate projected costs for an MVP within a fair range, and the consequence of inaccurate estimates is mitigated as the overall timeframe shortens.



Reconciling continuous delivery with IT governance

Cloud and containerisation have catalysed an explosion of CI/CD tools which are moving responsibility for the full software lifecycle into the development team's hands, under the labels of Agile, DevOps and automation.

These new "pipelines" can deliver software directly to development, testing and even production environments, which is immediately at odds with the traditional change control processes incumbent to many large organisations. For good reasons, governance checks existing in this lifecycle to address release management, security, and infrastructure among others.

The challenge

We want to break down these governance silos, with legacy thinking that puts the brakes on agile delivery, but we still must deploy software with confidence, and with discipline.

Bold steps are required to transform into a true DevOps mindset.

The first of these is working to collapse the silos which are disrupting an agile workflow, methodically re-deploying staff from governance-only teams directly into the delivery teams themselves, promoting mixed skillsets.

Retain critical control gates while adjusting to a continuous delivery model. It is completely feasible to automate 90% of the process in a way that still leaves final control over major deployments in human hands, while removing most of the human error.

A culture, not a team. It can be easier, but counter-productive, to re-skill an existing release management team towards new cloud tooling and then label them the "DevOps team". This is an anti-pattern, and defeats the purpose of creating teams who are collectively responsible for both building and operating their software.

Cross-functional teams

A cross-functional team is self-sufficient.

They can resolve their own blockers, and make most decisions internally. They can clearly see and measure their goals and have all of the tools and agency required to take steps towards them.

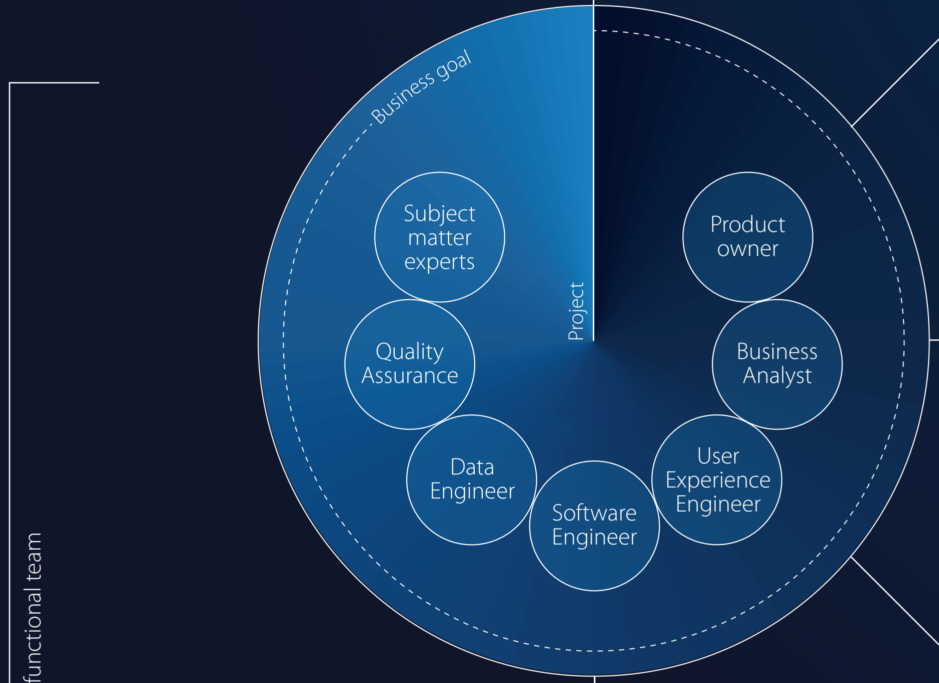
To achieve this, each team must include a blend of skills, knowledge and decision making within its own ranks.

Autonomous teams

We need autonomous teams with the tools and accountability to work as independently as possible. This greases the wheels of delivery, and removes traditional dependencies and the soft excuses which result in delays.

Joint ownership

Each team should ideally share a high degree of joint ownership and responsibility for the results, with access and visibility of the impact of each milestone. A source of motivation, and clear targets.



Eliminate hand-offs

We're working to find and remove "hand-offs" between teams. Artificial processes which create distance between roles. For example, from business strategy to analysis, then again to a different development team, before being handed off to testing.

Faster turnaround

The less waste in the process the better as the turnaround time from idea to technical manifestation and review is reduced, ultimately saving money.

Business participation

We are seeing that a culture of business participation is a growing trend within successful projects. Product owners and subject matter experts need to be encouraged and expected to attend and engage in the development processes. And the tech team reciprocates, with a vested interest in strategy and product.

Overlapping skills

This multi-disciplinary model creates opportunity for overlapping skills to benefit a product. Analysts can code, product owners can test or design. All available skill and capacity in the team is taken up all of the time.

Talent matters

It should go without saying, but the level of talent within the team executing on a project will have a dramatic impact on both the quality and the usefulness of the resulting system, regardless of how well specified it is.

A large amount of work exists in between the lines on every project. There is unavoidable room for interpretation in every user story.

These gaps are where **individuals within a team will rely on their own critical thinking and contextual understanding to make the best decision**.

Critical thinking

This process of filling in the gaps relies on experience, smarts and problem-solving willpower to create usable and quality systems. Without adept thinkers, poor decisions in these spaces can add up to major issues on a feature or product.

Highly technical roles in the software delivery process are not commoditised, although they are sometimes seen that way because the work can be invisible.

Engineering details will make a substantial difference to the resulting product and it is not realistic to specify these upfront (or to constantly stop and resolve small issues through a governance process).

We are reliant on people to identify, own and solve for these issues with high frequency for great results.

Ultimately, beyond just pace of delivery, we need to **place more value on the initiative, ideas and problem solving culture** within our tech teams or partners, so we can be confident they're able to represent business interests.

For practical ideas to recruit and retain your own top technology talent, read our publications.

Click through to read these publications ↓

The adoption problem

Internally deployed systems are frequently rolled out to users who resist the change they bring, working around them.

Modern change management requires deeper participation in solution design and the agile process simultaneously invites this. We need to make development a more inclusive and consultative process to get ahead of the adoption problem.

There is an easy opportunity to start cultivating "change champions" by **including them in the development processes early on**, gathering their feedback and allowing them to guide priorities. These people then organically spread the work and create a groundswell of positive interest and buy-in for the upcoming systems.

This also **creates ownership, purpose and responsibility** within those who are selected to participate in the project. These people then advocate for the system, changes and processes that it comes with, without relying on a centralised rollout.

We see this making practical sense too, as these champions are usually subject matter experts whose contributions will make sure that a system will be fit for purpose.

The human element is also important. People impacted by technology need to feel consulted, and will be more interested and willing to engage with it in future.

This is not to advocate for design by committee, which simply bloats scope. It is to **invite practical participation which has often revealed quick wins that may have been unseen**. The stance here is important.

Responding to failure

What to do if you didn't get it right the first time, and now need to leverage what you have as best as possible to move forward.

A If your solution doesn't fundementally solve the problem it was intended to

Retrofit and modify
Simplify the problem scope and therefore system scope to create a new target around which to proceed with this thinking.

B If you're confident in your solution, but adoption is poor

Late-stage change management
Technology can change processes and roles in a way that will be met with resistance. If you're too late to soften resistance prior to launch, then you need to get creative with how to roll your systems out.

C You've over-invested in a solution that still isn't finished

The frustrations of a sunk cost with no return in sight
Blindly pursuing old promises is likely to make things worse. It's time to pause. Reign in spending and consolidate. Then take a moment to reflect, weigh up options and make a call on what direction to take.

Finding a way forward is complex, and will be influenced by business pressures, budgets and the size of the ship to turn. If you'd like help, reach out to us to explore:

1. Deciding whether to improvise on a project, or scrap an investment and start over
2. Drawing a line in the sand, revisit business needs and rapidly establish a way forward
3. Technology audits to fully quantify the extent of a technology or adoption problem

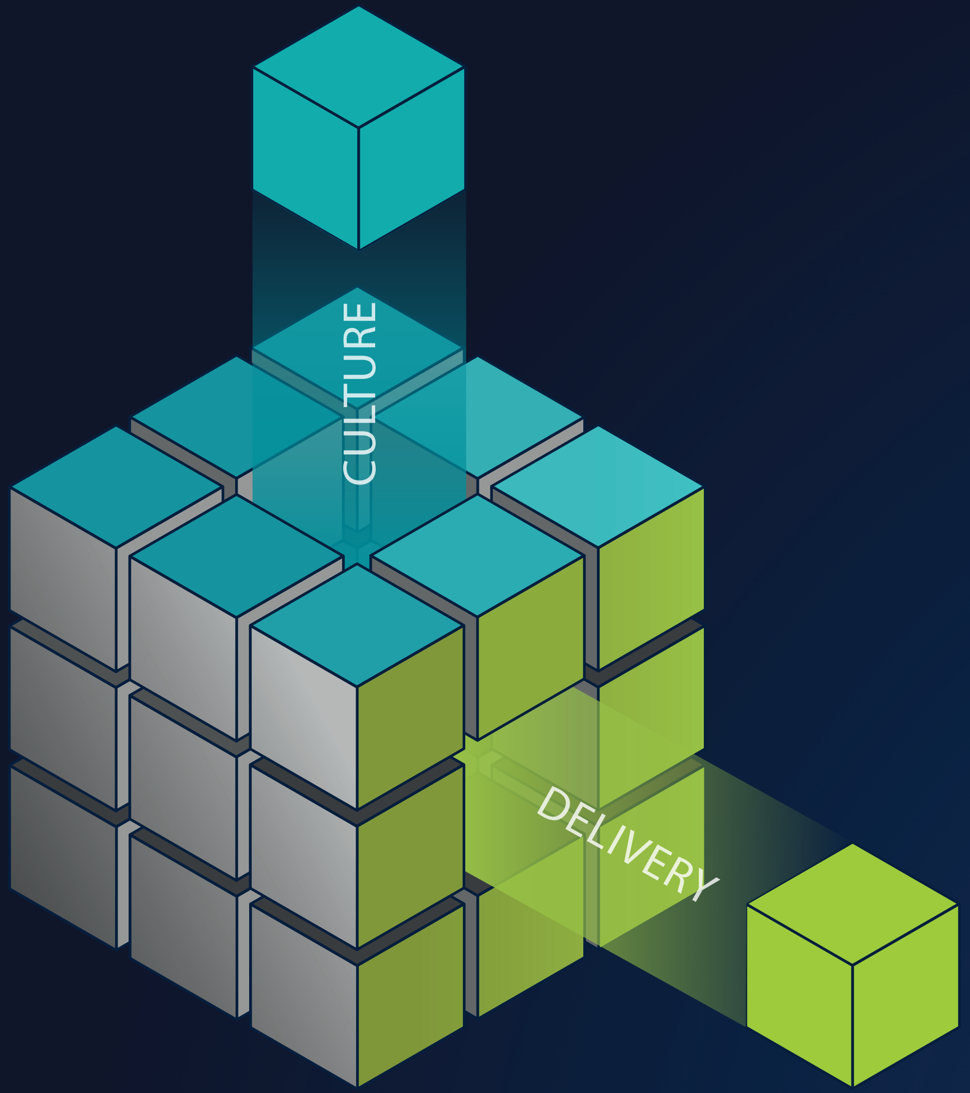
solutions@entelect.co.za ←

START THE CONVERSATION

solutions@entelect.co.za

We have been building and running diverse technology teams for more than 20 years at Entelect. We'd like to share these experiences crafting a culture that draws in the best possible talent across all disciplines, and leading large teams to real business results through skill, meaningful processes and critical thinking.

Shape your business for the future.



Recruitment
Magnetising your EVP
Finding the right candidates
Smart interviews
Onboarding

Retention
Culture and community
Practical tools for growth
Leadership development

Working remotely
Remote Conversations
Decision-making manifesto
Keeping culture alive

Problem Solving
Facilitated design sprints
Upfront versus continuous analysis
Co-design and collaboration

Methodology
Flexibility in process
Context, conduct and awareness
Prioritising business value
Communicating with business language

Quality
Pragmatism in practice
Shared expectations of done
User adoption and buy-in

entelect.co.za