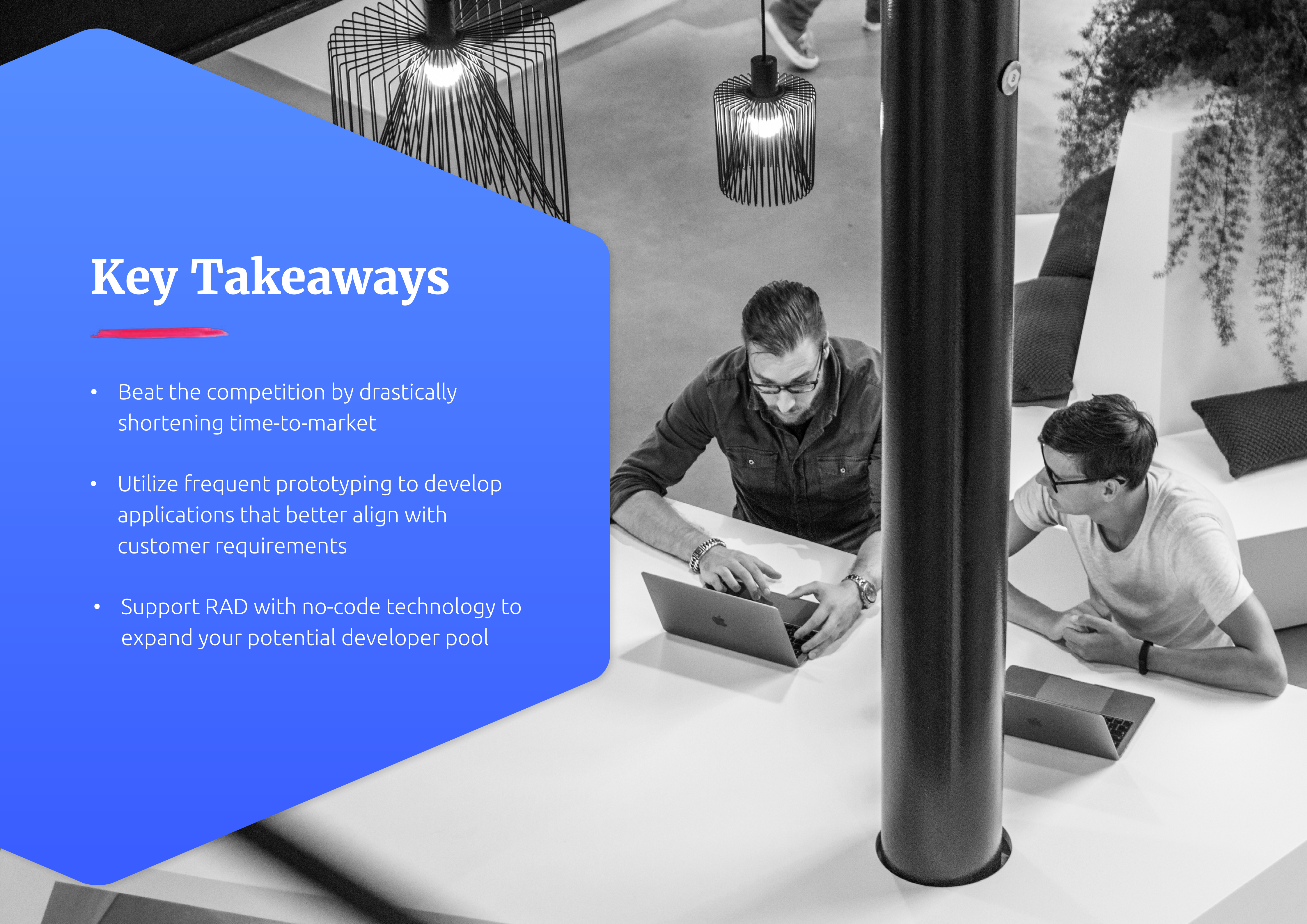
Betty Blocks

An Introduction to

RAPID APPLICATION DEVELOPMENT

Key Takeaways

- Beat the competition by drastically shortening time-to-market
- Utilize frequent prototyping to develop applications that better align with customer requirements
- Support RAD with no-code technology to expand your potential developer pool



Intro

Every sector and every organization has its own unique challenges. But when it comes to software development, there are some challenges that don't discriminate. Do any of the following sound familiar?

- Massive IT backlogs;
- Too much time and budget allocated to just "keeping the lights on";
- Poor alignment between business and IT;
- Slow time-to-market;
- Products that fail to meet customer expectations.

In this whitepaper, we'll look at how the more traditional software development lifecycles (SDLCs) could, when used in the wrong situation, be to blame for some of these challenges. And we'll look at how rapid application development (RAD) could be a more effective methodology for organizations looking to tackle their challenges, get ahead of the competition, and drive innovation.

You'll learn what RAD is and why it came about. We'll look at the kinds of situations in which RAD strategies should be employed for maximum effect, and we'll compare the RAD methodology to a widely-used traditional SDLC, the waterfall model.

With a sound understanding of RAD, you can make an informed decision as to whether it's the right model for your goals. That said, let's begin!

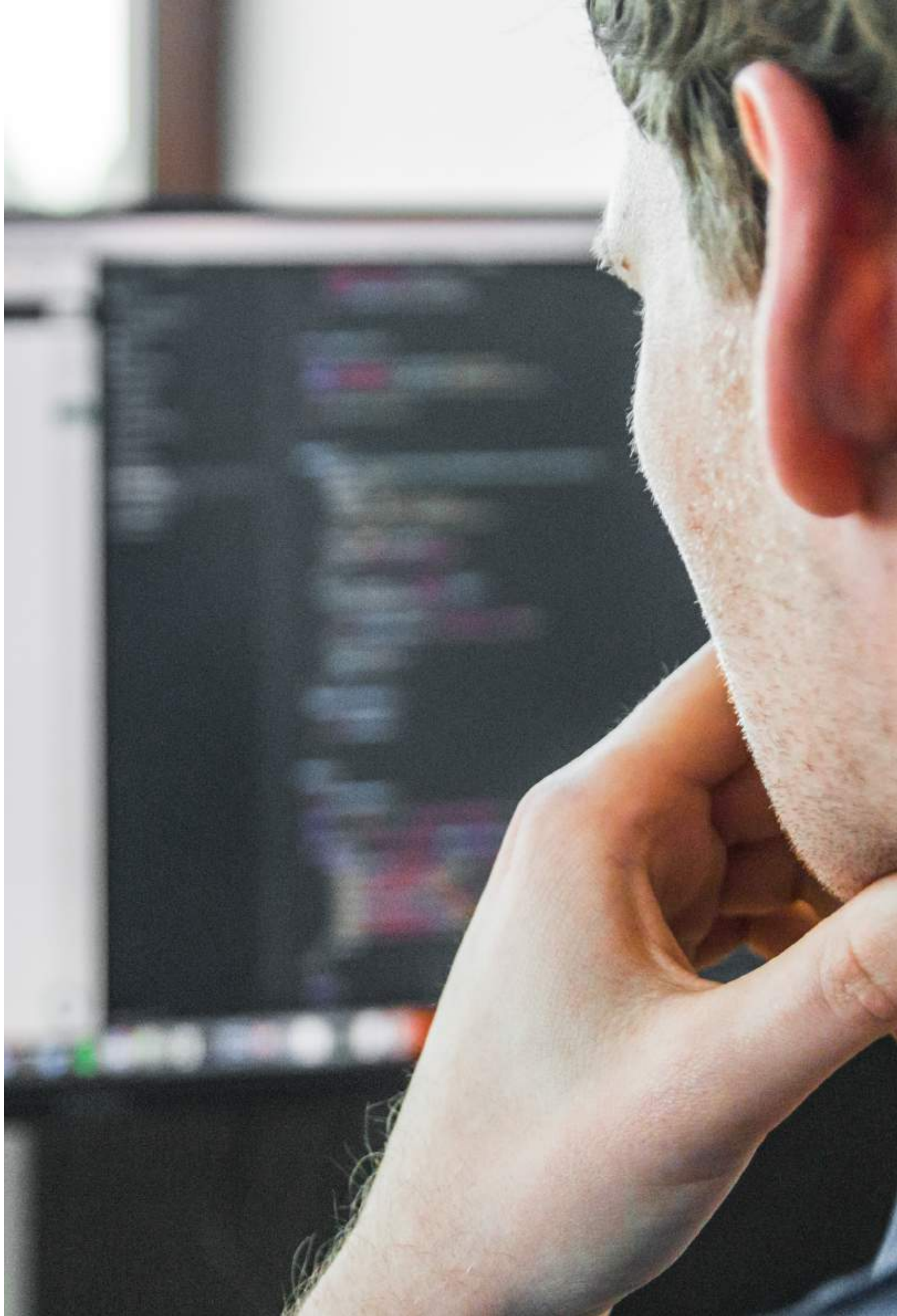


Table of contents

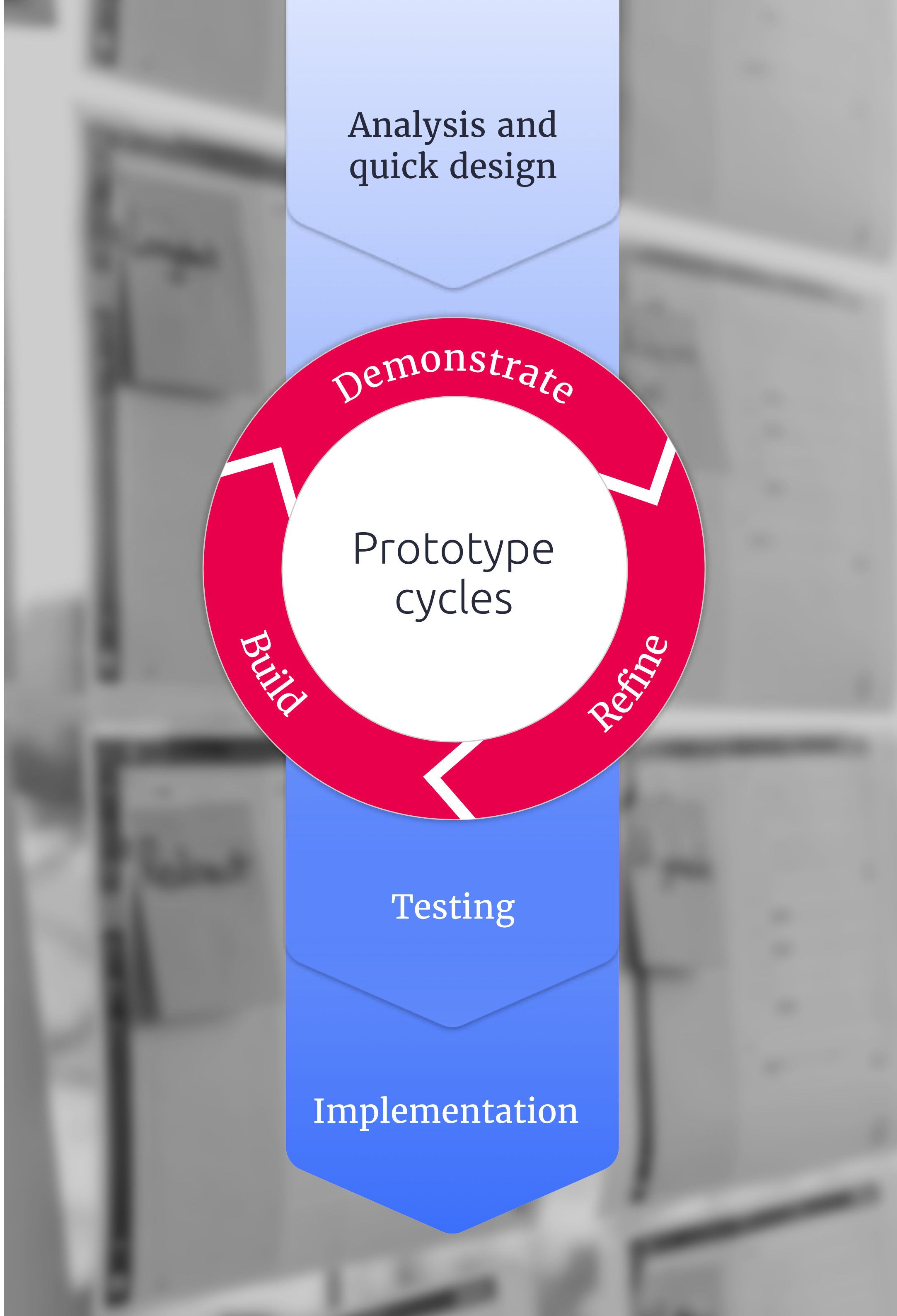
What, why, when, who?

What is rapid application development?

RAD was first termed way back in 1991, in James Martin's book of the same name. Here's how Martin described the methodology:

"Rapid application development (RAD) is a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle. It is designed to take the maximum advantage of powerful development software that has evolved recently."

On a basic level, RAD is a combination of Computer-Assisted Software Engineering (CASE) tools and a different way of thinking about the process of software development. In contrast to many traditional SDLCs, such as the waterfall model (which we'll examine in the next section), RAD is a more cyclical, agile methodology. Here's how it looks:



The 4 fundamental phases of the RAD model

1. Set project requirements

This is when stakeholders work together to ensure everyone is on the same page in terms of the goals for the project. What's key, however, is that with RAD, the planning stage is minimal when compared with traditional models. Rather than spending weeks or months defining every specification, stakeholders make more of an outline of the requirements. Contrary to traditional SDLCs, this is on the basis that specifications are refined and feedback implemented as the project moves along.

2. User design

This is where the RAD methodology really shines. Users get a feel for the product to ensure things are moving in the right direction. The design comes together through multiple prototypes, and with continuous user feedback, bugs and defects are fixed, and the product improves with each iteration.

It's also important to remember that your customer's business needs may change unexpectedly. With RAD, it's easier (and more effective) for customers to communicate how prototypes should be modified to accommodate such changes.

3. Construction

This is where prototypes become working models. Most glitches and changes were addressed during the user design phase, and so teams are able to complete the final product much sooner than via traditional SDLCs. The construction phase also includes testing, and it's still not too late to implement new customer feedback.

4. Cutover

This is somewhat similar to the implementation phase of traditional SDLCs. Everybody is happy with the product and it's time to launch the product into a live environment. Developers focus here on stability and maintainability. Documentation is written and user training is provided.

RAD in summary

Less planning, more prototyping

Whereas traditional SDLCs — the waterfall model included — focus heavily on the planning phase, RAD places emphasis on prototyping. This makes it incredibly agile. Why? Because RAD allows for continuous iterations, with developers making adjustments throughout the development process.

In today's highly competitive environment, it's crucial that the customer comes first. RAD enables customers to get hands-on experience with the product before the end of the development process, meaning feedback can be implemented earlier and more frequently.

Speed

Whereas traditional SDLCs require lots of manual coding, RAD has become associated with technology that enables the reuse of code, namely low- and no-code platforms. The use of such platforms as part of RAD strategies results in faster development times, as well as fewer errors. We'll look at low- and no-code platforms in more detail later.

Why do we have RAD?

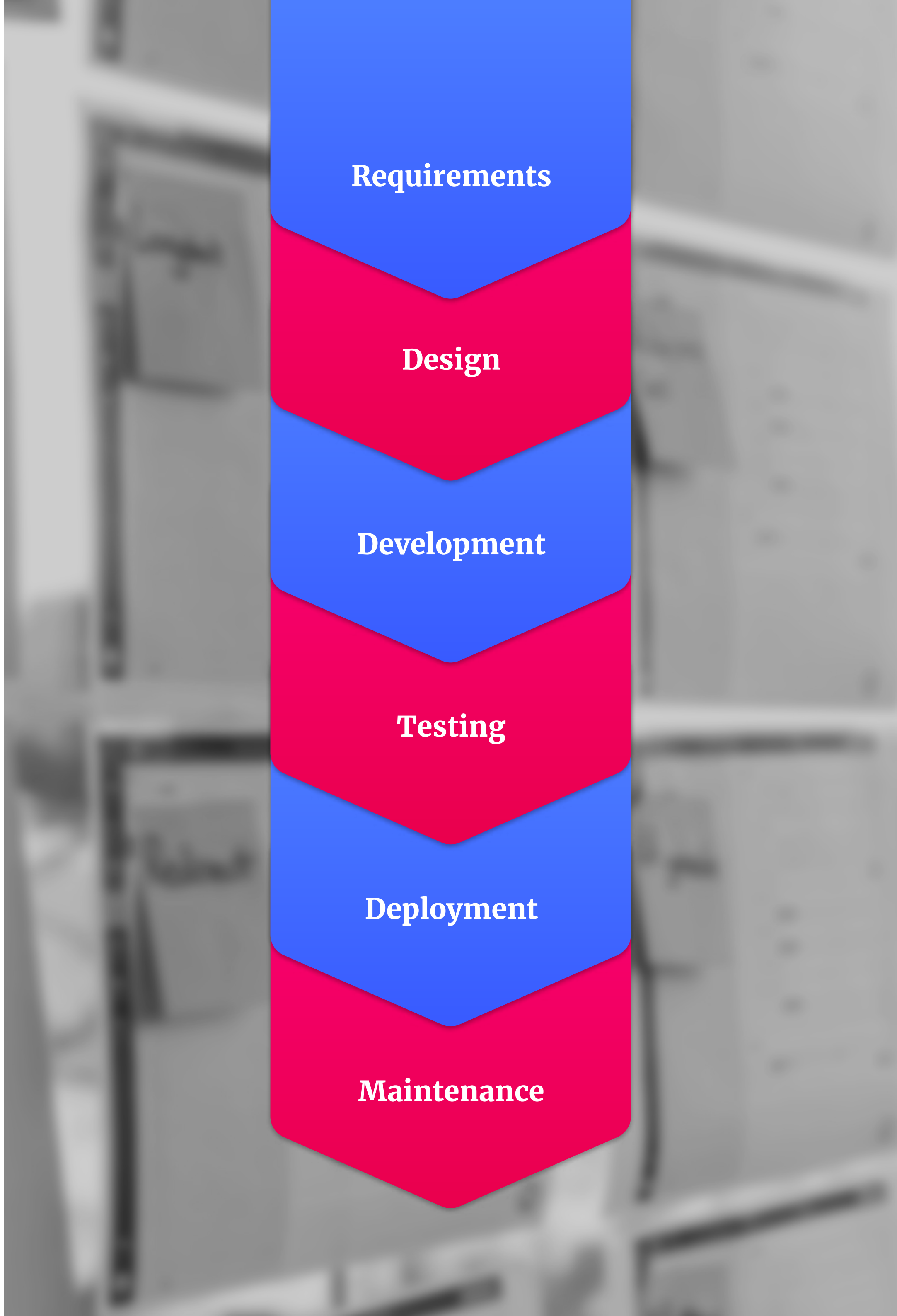
To better understand RAD, we need to look at why we have it in the first place; what was the need that led us to it? What was the world like before RAD?

The waterfall model

The RAD methodology, along with its predecessors — DuPont's Rapid Iterative Product Prototyping (RIPP), Gilb's Evolutionary Life Cycle, and Boehm's Spiral Model — all aimed to make the software development process more effective, whether by reducing risk, improving quality, or increasing development speed.

To understand why we needed a more effective SDLC, it helps to draw a comparison with a traditional SDLC: The waterfall model. The waterfall model follows a linear trajectory; each stage follows on from the last, flowing downwards like a waterfall.

The waterfall model dictates that one phase must be completed before moving on to the next, and each phase must be completed sequentially. It has an in-depth planning stage and there are few surprises. In this sense, it is somewhat more straightforward than other SDLCs.



Advantages of the waterfall model

Though it's been around since the 70s, the waterfall model is still widely used today. In its survey, Pulse of the Profession 2018, the Project Management Institute asked organizations what type of approaches they used for completed tasks within the previous 12 months. A whopping 47% said they used a predictive approach (such as the waterfall model). Why?

The waterfall model can be effective if projects have fixed timelines and budgets, and when there is little chance of the market and project requirements changing. The waterfall model enables organizations and customers to have clear visibility of the development process; each phase is laid out up-front and signed-off by stakeholders.

Pros

- Clear structure;
- Specific deliverables for each phase;
- The project moves along at a manageable pace (for developers, at least);
- Resource allocation can be easier due to the rigorous planning involved;
- May be preferable for customers who need a fixed start and end date for the development process.

However, problems arise when the waterfall model is the default, when it's used without much thought as to whether it's really the most suitable methodology.

Disadvantages of the waterfall model

One of the biggest draw-backs of using the waterfall model is that the prototype and development process must finish before the product can be seen, felt, and tested. Customers are left out of the process, only seeing the product after months of hard (and expensive) work has already taken place. Organizations risk reaching the end of a lengthy development phase, only to see that the end product fails to match the customer's requirements. Because so much rests on the initial planning phase, there is no room for misunderstandings or miscommunications, which you could argue are inevitable.

Moreover, what if requirements change? What if, 6 months into the project, the initial product is no longer viable? The inability of the waterfall model to incorporate changes and customer feedback can have huge implications for the future of an organization. Especially in industries that are fast-paced, when market changes can occur at any time, organizations should think carefully before defaulting to the waterfall model.

In today's tech-driven world, things change. Goal posts move from one day to the next; the needs of your customers change; the competition changes tactic; your main source of revenue becomes obsolete overnight because of a new technology.

Cons

- Unable to respond to changes and feedback until the latter phases;
- The customer is left out of the development process;
- Development times are lengthy;
- Emphasis is placed on documentation, rather than the product;
- Pre-defined requirements mean less scope for creativity;
- Potential to be more costly and risky than other methodologies.

The waterfall model and the FBI

Though it's not likely to come up in the next pub quiz, you might be interested to learn of a famous use-case of the waterfall model, one that ended particularly badly. In 2000, the FBI set out on a project to develop a system called the Virtual Case File. The development process lasted 5 years before the whole project was written off as a failure. The estimated cost? Over $170m! Though there were many factors that undoubtedly would have impacted the project's failure, the waterfall methodology almost certainly played a substantial role.

Had the FBI chosen an iterative approach such as RAD, they would have been able to respond to changes and feedback much earlier in the process, rather than realizing after 5 years that they'd missed the mark. Frequent prototypes would have confirmed whether or not they were on the right track.

This isn't to say that the waterfall model should be avoided at all costs. It merely serves to highlight the risks of the traditional methodology.

It's because of such risks (and such failings) that RAD was created. Developers need the ability to respond to feedback, market changes, new challenges, and new opportunities. We needed another option, rather than having to wait for the entire development process to conclude before feedback could be given and acted upon. Who has that kind of time these days!

When to use RAD

When a fast time-to-market is essential

We all know how competition has been steadily increasing across pretty much every sector over the last few years. A fast time-to-market is essential for many businesses; sometimes it's the only way to stay ahead of the competition. When the customer needs a prototype in a relatively short amount of time, eg. 2-3 weeks, RAD makes perfect sense.

When your customer is willing to commit to the process

Customers must commit to being involved in the process, meeting with developers for regular feedback sessions. If the customer can only commit to being involved at the start and end of the development life cycle, then you're not going to get the most out of RAD.

When you need to onboard new developers quickly

Because RAD emphasizes the reuse of code through using low- and no-code platforms, new team members can be onboarded quickly. There is no need to hand over huge amounts of code!

When you want to focus on the product

To enable your team to focus more on the product, using a platform that enables automatic updates across all applications can be hugely beneficial. This is achievable as part of your RAD strategy with a no-code platform like Betty Blocks.

Who benefits most from RAD?

Organizations

Organizations in fast-paced sectors

If your customer operates in a sector that is fast-paced, where changes can (and do) happen at any time, having the ability to adapt the product is crucial. You could start out with a perfect plan, with every requirement checked and rechecked, everything clearly communicated and everybody happy. But what good will it do if a competitor suddenly releases a new product that completely changes the game? Technology has increased the pace of pretty much everything these days, and how an organization responds to change can either make it or break it.

Organizations with strong teams

For RAD to work well there must be a high level of teamwork between all developers and stakeholders (and customers). Because of the fast-pace of RAD and the condensed planning phase, clear communication is vital.

Organizations in which shadow-IT is prevalent

Shadow-IT is a major problem for organizations across many sectors. Put simply, it's when business-users are in need of a solution, and when IT fails to provide it, they build it themselves. This has all sorts of implications, including:

- Increased security risks;
- Lack of standardization;
- Lack of internal support;
- Software upgrades that damage other areas of the organization's IT infrastructure.

One of the main reasons IT fails to provide the solutions needed by business-users is that, with traditional development methods, it simply takes too long and the resources just aren't there. With RAD, however, the speed of development (and close alignment between IT and business) means that the answer to shadow-IT can be just around the corner.

People

Pro-coders and IT

Pro-coders are vital to both the waterfall model and RAD (or any SDLC). The way pro-coders benefit by using RAD is by continuously receiving feedback and having the ability to implement it. By doing this, pro-coders know that they're on the right track, and that they're not spending months writing reams of code for a product that is way of the mark.

No-coders and citizen developers

It's typical to use a low- or no-code platform as part of your RAD strategy because of the increased development speed offered by such technology. But low- and no-code platforms offer more than speed. Perhaps your organization has already tried to recruit more pro-coders, only to experience the developer shortage first-hand? The benefit of using a no-code platform is that it gives organizations a whole new pool of talent to utilize: No-coders and citizen developers. The collaborative nature of RAD positions it as an ideal methodology for all 3 developer types to work together.

Business-side stakeholders
As someone on the business-side, you're best positioned to know the product requirements, whether the product is being developed for internal use or for a customer. You want to be sure that IT has understood all of these requirements, and that the product is coming together in the right way. The iterative approach of RAD means that the process is transparent; you're able to test the product frequently and see your feedback implemented.

Top 3
Pros and cons of RAD

Pros
- Adaptable and flexible: Respond to changing requirements;
- Better alignment with customer needs (even when the needs change);
- Speed: Faster time-to-market.

Cons
- Projects need to be large enough that they can be modularized;
- Strong teamwork and communication is needed, which may be harder to achieve for larger projects with lots of people involved;
- Customers must be committed to being involved in the whole process. This will be tricky if the customer expects you to just 'get on with it'.

RAD, low-code, no-code, and citizen development

Low- and no-code platforms compliment and facilitate the RAD methodology because they naturally enable faster development from a technological point of view. Because low- and no-code platforms are based on the reuse of components, developers are able to release frequent iterations, whilst not losing the ability to customize components with code when required.

Whilst both low- and no-code platforms support RAD, which of the two platforms you chose depends on your organization's resources and needs. For the purposes of RAD, both strategies are sound. The main difference with no-code platforms is that they allow business users to take part in the development process, because a high level of programming expertise isn't a prerequisite. This is what's known as citizen development.

Instead of merely helping facilitate communication between customers and developers, business-side employees play an active role in the development process. By doing so, they bring invaluable insight to the development process in terms of their knowledge of the customer's needs.

The key thing to remember about no-code and citizen development strategies is that, in order to be successful, they must be properly governed. It means that pro-coders will always be essential, but they're no longer restricted to simply 'keeping the lights on', and innovation can thrive.

Ideal platform specifications for RAD

Remember, RAD is a methodology. To get the most out of your RAD strategy, you need a platform that supports it. Here's a list of specifications we recommend should be part of your low- or no-code platform, in order to maximize the effect of your RAD strategy as a whole.

Model over code
RAD platforms should emphasize and support a model-over-code development strategy; they should include a model-centric UI layer designer that enables basic CRUD application designs with little or no coding. Some advanced RAD tool-sets provide visual drag-and-drop process-flow creation with data and UI integration, and application analytics.

No-code support
While the emphasis with RAD platforms is model-over-code, this doesn't mean "model-at-the-exclusion-of-code". Rather, RAD platforms often allow the front-end developer to hand-code when necessary. To this end, these tools should provide a high-level scripting language (either proprietary or using an OSS-based technology).

One-button application deployment
RAD tools should support automated application deployment from development environment to production. They should support staged deployment activities, such as development to testing, testing to model-office, and eventually to live production.

One-button revision control
RAD tools should support a single, managed view, with point-in-time revision control for all software assets in the development project.

Live prototyping
RAD tools should provide built-in support for wireframing and prototyping integrated directly into UI design.

Runtime platform support
RAD tools may support multiple runtime deployment platforms, including on-premises, private cloud, and public cloud. However, it's not uncommon to find these tools already built into a specific runtime platform today.

Service-oriented architecture (SOA)
RAD tools should support a service-oriented architectural foundation. As a minimum, these tools must allow developers to consume IT services from external providers, typically through REST (and potentially SOAP) interfaces.

Smart application connectors

While RAD tools must support low-level external web services, they should also place a priority on ease of use and developer productivity.

Collaborative development

RAD tools should support team collaboration features such as task/to-do management, code review and markup, messaging, and reporting.

Reusability

RAD platforms should allow developers to share application assets such as templates, services, components, forms, workflows and so on. These tools should support in-house social collaboration.

Betty Blocks and RAD

How does Betty Blocks support the RAD methodology? As we've seen, no-code platforms and RAD strategies are like wine and cheese: an ideal pairing. The business side is best positioned to understand the needs of the customer, and a no-code platform like Betty Blocks creates better alignment between business and IT. It brings everyone that should be involved in the development process together and allows them to continuously review and refine the product.

All 3 types of developer are utilized with a no-code platform: The pro-coder, the no-coder, and the citizen developer. Each has a unique and vital role in executing a successful RAD strategy, enabling faster development, prototypes in days instead of months, time-to-market in months instead of years, and applications that actually exceed customer expectations.

About Betty Blocks

As the world's leading no-code platform, Betty Blocks empowers both enterprises and Citizen Developers to build complex applications efficiently and effectively without writing a single line of code. With its focus on people, Betty Blocks empowers organizations to work towards the right solution and enable the workforce to take control of their innovations. Cloud-based Betty Blocks is available worldwide. The company has offices in the Netherlands, Belgium, Germany, US, Mexico, UK, Japan, and South Africa.

Visit us at www.bettyblocks.com and follow us on Twitter and LinkedIn.

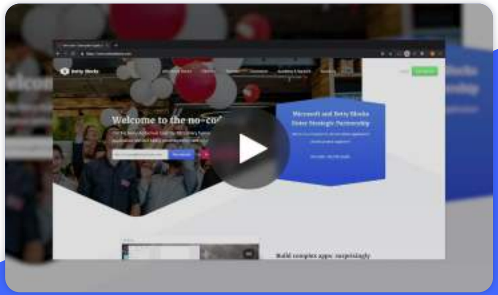
Greetings from the team!

Ryan Whitmore

Content Marketer at Betty Blocks

By now you know why no-code is the way to go

Want to find out more about the Betty Blocks no-code platform and how it suits your business case? Discover our feature videos and platform demo right here.



Discover the platform

Want to know more? *Download our other whitepapers or visit the website*

The No-Code Solution to your Digital Transformation

Are you a CIO looking to make organizational change? Have you made changes that have proved ineffective? This is your guide to approaching digital transformation the right way.

Get the whitepaper

The Rise of the Citizen Developer

How we view software development — and what constitutes a developer — is changing. Embrace the paradigm shift and successfully implement Citizen Development with this guide.

Get the whitepaper

The Ultimate Guide to No-Code

What is no-code? With practical implementation cases from around the world, this guide shows you how no-code will change the way your business develops applications...forever.

Get the whitepaper

Governing Citizen Development

Citizen Development is set to dominate boardroom discussions within 5 years; a solid strategy will be vital to success. Get a head start on the competition with this guide.

Get the whitepaper

The Developer's Manual of the Betty Blocks Platform

This no-nonsense guide shows developers and IT professionals where the magic happens, with an in-depth look at the technical elements of the Betty Blocks platform.

Get the whitepaper

Gartner

Gartner Magic Quadrant for Enterprise Low-Code Application Platforms

Betty Blocks named a Visionary in Gartner's 2019 Magic Quadrant for Enterprise Low-Code Application Platforms. Read the report to find out why.

Read the report