UNROOLE FOR AGENCIES

- Faster turnaround and iterations
- No need for backend operations
- Less resources needed for development
- More integrated decision making process

unroole

A better way to **build the web.**

You're looking to build a family of networked websites; ones with similar content, code, and structure. With most of the solutions available on the market today, this means setting up a new instance for every website. Clearly this doesn't scale well and requires a lot of redundant effort. You'll probably want the ability to stage changes before sending them to production, which means setting up even more instances along with migrating content and code between environments. On top of all of this, most systems make your content and developer teams work within strict, templated confines.

With unroole, we've radically changed the web development workflow to avoid these common pitfalls. Multiple websites can be managed within one account, allowing for shared content and code. Staging environments are rendered obsolete with the ability to schedule and version all code and content, as well as the ability to preview your entire website with yet-to-be deployed changes. Lastly, unroole's approach allows developers to create entirely custom building blocks for content teams to use. No confines of default templates.

In this case study, we discuss in further detail exactly how unroole shifts the web development paradigm to a more modern approach.

Boban Vejin

CEO & Co-Founder

What is unroole?

unroole is a tool that **cuts down man-hours** for web development and administration by roughly 60%. This is achieved by intelligently harnessing the power of the cloud to enable you to host an unlimited number of websites with maximum scalability and never before seen flexibility of networked websites.

Simply put, we have removed all the redundant and repetitive tasks from web development. **You are left with more time to focus on what makes your websites unique, and not waste time on what makes them the same as other websites.** unroole gives you the opportunity to provide a world class service and establish firm relationships with your clients.

unroole in an agency workflow

Faster turnaround and iterations

"unroole is enabling faster website development turnaround time of both entire project and iterations in code, structure or content."

In your current workflow, after your account **agrees to a brief** with your client, the work on a new website starts with UX and development teams defining functionalities needed and creating a visitors workflow most suitable for the goals defined in the brief. After **the wireframe is made** and signed-off by the client, it is sent to designers. Designers create **several starting points** collaborating with the UX team and the client, through an account, and implement their feedback into **the final design**. After the design is approved, it is sent to the development team so they can start work on **backend and frontend code**. They set up a **local environment**, develop the website, and test it with the mock content. After internal QA and fixes, the entire environment, codebase, and mock content are **migrated to a staging server** so the client can preview the work. After the client's feedback is integrated on the local environment, changes are **again**

migrated and overwritten on the staging environment. The client adds real content on the staging site, which your team then **migrates to the live server**. Congratulations, job done!

But, we all know that things never go this smoothly. This world basically exists somewhere over the rainbow. The reason for this is that the whole **process** of decision making and subsequent work is **linear**. Each milestone described above depends heavily on the decisions made in the previous milestone, and if, in the waves of hindsights, there are changes to be made at any point in the process, **your team has to take a few steps back and rewrite work done in the previous milestones**. If in the middle of the development process your client sees the first results of your work and realizes that a different homepage structure would be more suitable, you have to go back to the UX stage to determine what is the best way to implement this change. Then, the designer creates a new version of the homepage, gets client feedback and approval, and sends it to the developer team. Again, from the local environment, changes are migrated to staging and then to the live environment.

Another common pain point in this workflow is when real content breaks the structure and layout designed using mock content. Again, the process is pushed back to the initial milestone. Often, **this is the case for every iteration,** including those which happen after the launching date. To make matters worse, most of these iterations **can't be anticipated** and planned in the price setting stage. This can severely **damage your profits** if you are spending more man-hours.

"unroole implemented agile approach to the very tools of web development which enables your teams to efficiently iterate without stepping on each others toes."

We are introducing the **separation of concerns between structure, code, and content**. This separation gives your teams the opportunity to work independently on their changes and create various versions of a website by combining versions of code, design, structure and even content. **With unroole, you can simultaneously create versions of each of these elements, then assemble them as lego blocks in a website preview**. None of these versions are live until they are signed off, and because of the modular approach, combining changes of all iterations is possible with just a few clicks.

unroole saves tremendous amounts of time during the iteration process by putting the whole project as a centre of collaboration between various project teams and the client, thus **avoiding the bottlenecks a traditional workflow causes**.

No need for backend operations

Currently, every time you are building a new website your development team has three environments to set up and maintain - **local, staging and production**. In these environments they are setting up separate databases and developing server-side code for your clients website to retrieve and store data. Every time a build needs to be demonstrated to a client, **all the content and code has to be migrated to the staging environment**. What is often the case, in order to save time, the client is adding real content to the staging environment and adjusts the configurations and options from the admin side. In order to preserve this work, your development team needs to spend **more time to merge the differences** between local and staging environments so work won't be lost.

"unroole is removing backend and server operations from your queue."

This constant merges and migrations between environments get even more complicated once the **production environment gets into the mix**. Quite often, there are changes to be made after the launch date, as usability data you get from the real visitors doesn't always match perfectly with your plan. Now your development team has to migrate production to the local environment, make changes, migrate to staging, and then migrate back to production. If you are working on an integrated marketing campaign for your client, and have several websites which serve different purposes in the campaign (landing pages, micro-sites, sign-up funnels, etc.) you have to **replicate this process for each of them, and duplicate the codebase and content you are using**. This process is a huge drain on your resources, but the actual work that is preformed is always the same, for every single website.

"unroole redefined the workflow by automating repetitive and redundant tasks of setting up environments for development."

Backend operations for creating and maintaining each website are done by unroole - **all iterations in code, structure and content are done in their own versions, and then assembled into a preview**. This is how unroole removed migrations

from the workflow. All versions are in the same place, and instead of migrating them to different environments, you just assemble different previews and publish the winning one.

This way of handling iterations is proven to be indispensable once the website is live, as your **clients can make changes in the structure of webpages on their own**, safely without the risk of breaking the layout or waiting on your team to make the changes.

Another use case where unroole outperforms other platforms is when your client needs a **network of websites that should have the same code and content base**. This is usually the case when a promotion strategy depends on cross-channel content distribution, and you have separate website channels targeted at different audiences.

"In unroole content and code are reused, not shared."

One of the core unroole features is **content and code reusability among websites in your account**. Usually in multisite setups, code and content are shared, not reused, thus process of adapting individual websites is **not flexible enough** and requires a lot of leg work. In unroole, the structure of websites is separated from the actual code of the website so **code can be reused in a modular and scalable way**, by just assembling pages' structure with different blocks of content. So, in order to create several websites with the same theme (to preserve the visual consistency of the campaign), all you have to do is make a few clicks in your dashboard and assemble pages of new website with a drag-and-drop interface.

Less resources needed for development

"unroole website themes are built with only HTML, CSS and JavaScript"

Today's websites usually have some sort of CMS backing them enabling easy content publishing. To develop a website for any CMS, you need the backend work done in order to set up the server and database. On top of that, you need front end work done as well to implement the design and structure of the website. This means that **you need two different skill sets** in your development team, which increases your production costs and turnaround time.

When **unroole removes backend operations** from your queue, it also removes necessity do develop back end code that communicates with a database. All you development team is left with is **development of the front-end portion of the website code**. Fetching the data from unroole's CMS is done in the front end code. By moving all the coding to the front end, we have enabled developers to work in a **flat information architecture**. When programming an unroole website, the programmer is dealing with the data in JSON format, meaning that you have a **total control over how the data retrieved will be presented.** There is no need to overwrite the default formatting, as default format doesn't exist. **Unlike with other CMSs where you have to adopt and confine to architecture given, unroole is information-architecture agnostic.**

"unroole is information-architecture-agnostic"

To make theme code reusable and modular, theme files consist of code chunks for each element of the page, navigation, call-to-action sections, etc. This **reusability enables you to cut down development time** as nothing needs to be coded twice. All elements that define the structure of the page are coded only once and assembled in the page building process. Iterations and changes are implemented with the least possible work required.

More integrated decision making process

Linear workflow of **create - edit - preview - publish** is excluding everyone from the process except for the one person making the changes. These changes may be changes in content, page structure, or design, and they are more **relevant to different members of your team depending on their expertise**. Also, there is no way to see how changes will be **affecting the website in its entirety** without publishing them. One person can't possibly sum up all the knowledge and skills your entire team possesses. This is where most common bottleneck is - one person needs more time to implement all the changes.

"unroole makes it possible for different team members to be included in the decision making process."

Because of the linear process, there is **no opportunity for comparing different versions of the website.** In order to compare two different website structures or designs, you would have to **set them up as separate instances** and distribute the links to the decision makers - client, UX, designers, and developers teams. **This does not scale well**, especially when final solution is a mix of the two proposals. Also, changes are usually isolated from the rest of the website so you can safely preview them and not mess with the live code. The way this safety net is constructed is **preventing** you and other team members from having a **broader picture** of how changes made will affect the entire website.

unroole website design, code, structure, and content - can have **infinite versions**. These versions can be easily used in any combination to **create a preview of the entire website**. Previews can then be distributed to all interested parties via a password protected link. **This workflow is proven as indispensable when testing versions on focus groups, or A/B testing once a website is live.**

"unroole gives you an opportunity to make more informed decisions"

unroole gives you the opportunity to make more informed decisions, which are crucial for **creating user-centric experience** on your clients websites and, more importantly, **implement these decisions in the most efficient way.**

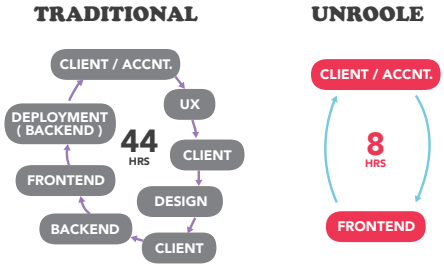
{un} unroole

A BETTER WAY TO BUILD THE WEB

unroole is a solution designed to build a family of websites in the most efficent way.

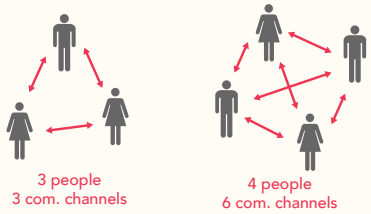
FEEDBACK CYCLE IMPROVED

Versioning content and previewing the future state of a website in *unroole* allows publishers a parallel workflow, resulting in faster feedback.



TEAM SIZE MATTERS

Larger teams create greater inefficencies. The number of communication channels based on team members(n) equals **n(n-1) / 2**

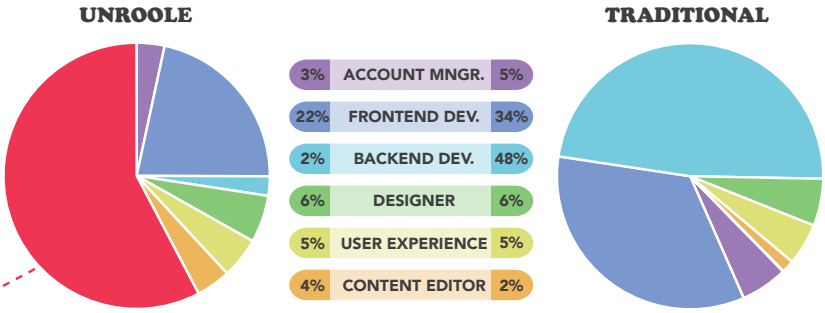


As communication channels increase, inefficiencies increase exponentially.

SMARTER ROLE DIVISION

By eliminating backend development from the workflow and optimizing remaining workflows, *unroole* is able to reduce develpment time by 58%.*

SAVINGS WITH UNROOLE **58%**



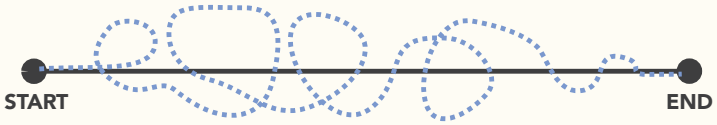
*Based on a family of 3 websites (approx. 100 pages, 100 articles, responsive theme).

POWER TO THE CONTENT PEOPLE

unroole allows content editors to use blocks built by developers, empowering them to create pages and make changes themselves.

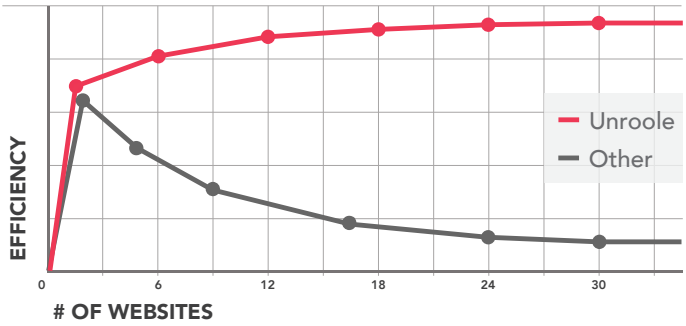
PLAN FOR CHANGE

Projects often go off-course, requirements change, and iterations add up. Separating content, code, and structure makes last minute changes less painful. What's your strategy?



EFFICIENCY THAT SCALES

unroole cuts the time of additional websites in half, allowing you to scale gracefully.

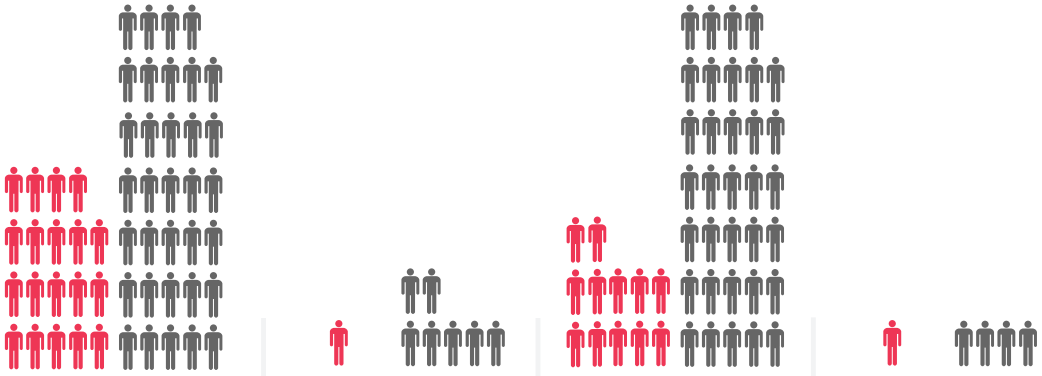


SAVINGS PER PROJECT PHASE

1 Resource Represents 7 Hours

unroole

Traditional Development



All unroole features are designed to save on time, the cost of developing, and the cost of managing a website. There is no need for backend developers, no need for migrating changes and content to multiple environments, and no need to create and upload content to several different websites.

The paradigm shift in website development and administration that unroole is introducing is about automating processes that have not been changed in the last decade by any of the available platforms or frameworks, open source or paid. unroole is giving more freedom for developers and more control for the content editors.

To start using unroole sign up for free at www.unroole.com/sign-up

To learn more about the unroole workflow visit www.unroole.com/learn

A better way to **build the web.**