SOLUTION DESIGN

BEYOND SCALE:
How to make your digital development program sustainable

Module 4 of 7
A technical solution doesn’t just involve software — it includes requirements, infrastructure, technical partnerships and the processes by which the solution is developed, updated and maintained. These components are likely to have evolved as you piloted your digital solution, potentially in different locations, and scaled it to a moderate number of users. But if you are now planning to either scale it significantly or supplement it with new digital services and products, you may need to make radical changes to your solution design.

One common evolution is towards simplification and stabilization of core code to enable replication and the ability to handle an increased volume of users. Another common evolution is the creation of new features, products, or services to enable diversification and the ability to reach new kinds of users or provide new services to existing users. Your path to scale and sustainability may require both.

This module draws on interviews conducted with digital leaders of organizations including BBC Media Action, Esoko, Akros, Dimagi, HealthEnabled, Vital Wave, BeeHyv, IMI Mobile, Echo Mobile and D-tree about their experiences designing solutions for scale.

This module will help non-technical NGOs and organizations that build software think about the unique challenges that digital programs face when scaling technical solutions, highlighting important technical considerations that may impact their sustainability. This
module includes key steps that address common questions asked by technical teams when planning for scale. However, each program is unique, and not all these steps may be relevant to your journey.

**It will help you:**

1. Use a human-centered design approach to reassess your user requirements
2. Consider external factors that may change your existing solution design
3. Prioritize new features and set targets
4. Plan your technical development for sustainability at scale
5. Enhance your development processes if you decide to build your solution
6. Deploy, migrate and maintain your scaled solution
Key steps

1. **USE A HUMAN-CENTERED DESIGN APPROACH TO REASSESS YOUR USER REQUIREMENTS**
   - Update your user personas
   - Update your user journeys
   - Consider accessibility and localization

2. **CONSIDER EXTERNAL FACTORS THAT MAY CHANGE YOUR EXISTING SOLUTION DESIGN**
   - Consider new system integrations
   - Consider updating your data model to accommodate scale
   - Consider how new regulations, policies or laws might impact your solution design
   - Consider streamlining your reporting requirements

3. **PRIORITIZE NEW FEATURES AND SET TARGETS**
   - Agree to target features for a minimum viable product
   - Work with stakeholders to get buy-in
   - When scaling to a new geography, budget additional time and effort for feature prioritization

4. **PLAN YOUR TECHNICAL DEVELOPMENT FOR SUSTAINABILITY AT SCALE**
   - Evaluate standard systems with professional support
   - Use requests for information or proposals for a structured, in-depth comparison of software options
   - Revaluate your hosting strategy
   - Assess whether you or your technology partner has the staffing in place to deliver

5. **ENHANCE YOUR DEVELOPMENT PROCESSES IF YOU DECIDE TO BUILD YOUR SOLUTION**
   - Reinforce your development team
   - Perform a code and backlog review
   - Revise your development roadmap or release plan
   - Build, test, modify and repeat
   - Consider the liability of open source vs. proprietary software and content

6. **DEPLOY, MIGRATE AND MAINTAIN YOUR SCALED SOLUTION**
   - Deploy and migrate (if needed)
   - Find the right time and frequency to release updates
   - Plan for ongoing operational costs
STEP 1

USE A HUMAN-CENTERED DESIGN APPROACH TO REASSESS YOUR USER REQUIREMENTS

Update your user personas

The size and complexity of changes required to your digital solution are partly dependent on the differences between your new and existing users. Updating user personas through focus groups with new users in different locations will help you understand whether your solution is relevant to them, and how it needs to change to meet their needs. If viable, build on existing user personas that you have already documented. Updating your user personas before making changes to your solution can save you a lot of rework later.

FOR MORE INFORMATION SEE:

► Business Model: STEP 1. UNDERSTAND WHAT THE CHANGING LANDSCAPE MEANS FOR YOUR DIGITAL SOLUTION

Update your user journeys

Mapping new user journeys is an essential analytical step for scaling. One technique for mapping new user journeys involves small group exercises where users representing the same persona work together to document visually how they would use the solution to accomplish a goal. This serves several purposes, including working with new users to validate their needs, helping you understand potential user behaviors and providing an early indication of new features that may be required. User journeys can be mapped from the perspective of potential new users — who describe how they would use a solution and what features it might have — or from the perspective of existing users — who describe how they currently use a solution, which can help identify current issues.

If you are planning to scale an existing solution without making significant changes, then testing the ability of new users to operate the solution can provide valuable early feedback on solution acceptability in the new environment. If new features or services are expected to be developed, paper prototypes of the solution can be similarly useful.
In practice | Esoko

Leading partners to create the best solution for users

The first thing we did when we decided to launch Tulaa in Kenya was make a trip to the field. We got everyone together, including our executives, and wrote down the key questions we needed answered by our potential new users. During our first field visit, we talked to farmers, state agronomy officers and managers from farmer cooperatives. Agronomy officers were especially important because they oversee a lot of farmers’ activities and provide insights about the big picture. At the end of each day our team debriefed back at the office and updated our assumptions, user personas and user stories.

We shared our updates with our partners, including the insurance company UAP and microfinance creditor Musoni. We learned from farmers that they wanted a longer period to repay loans, and Musoni agreed to create terms that would work for our new users. We also learned that many farmers didn’t have a Kenya Revenue Authority personal identification number (PIN), which is required by insurance companies to obtain policies. UAP worked with us to create a group insurance policy that our users could purchase at a reasonable price.

We brought this news back to the farmers to see what they thought of our new solution. Explaining what our solution offered took over an hour! Insurance policies legally require that individuals purchasing a policy fully understand what they are paying for and what they will get paid in different situations. We worked with both partners to craft a simplified and legally sound explanation of what our solution offered that could be explained in just a few minutes, and we tested and refined it during our next visit to the field. We then brought the refined text back to our partners and worked to create marketing materials that we knew would be well understood by potential new users.

The first thing we did when we decided to launch Tulaa in Kenya was make a trip to the field.
Consider accessibility and localization

Localization requirements for scaling a solution may vary widely depending on the size and diversity of the country. In many countries, you are likely to encounter substantial differences in language or dialect, level of education, digital literacy and even income — particularly between urban and rural areas. Replicating your service in a new location in the same country might thus be as complex as developing new products and services, requiring substantial changes to your underlying solution. For example, if you’re planning to replicate a solution in rural areas that you developed for urban audiences with smartphones and 3G access, you may need to move to a 2G audio-based technology, such as interactive voice response (IVR), that’s accessible from older model mobile phones.

In addition to supporting different media formats (audio, text, video), technologies (IVR, SMS, USSD, mobile data) and languages and fonts, you may also need to provide users or administrators with a way to select a specific version of the solution, such as ensuring users actively select or confirm their language preference before receiving information via IVR or SMS). These changes can be time consuming and costly to implement all at once, so it’s important to identify and prioritize those that are critical to launch and those that can be implemented later.
Other examples

- Listen to Benjamin Winters, Regional Director at Akros, talk about the impact of identifying and redesigning their solution for a new category of user.

Dimagi created a process for rapid localization of mobile-based multimedia content to enable standardized health messages to be translated by local health workers into local dialects in rural communities. The use of audio allowed new translations to be recorded quickly and cheaply in every deployment area, enabling the program to be replicated across a wide variety of culturally and linguistically diverse locations.

How to

- Read more about user journey mapping here and here
- Read more about creating user personas here and see some examples here
- See a quick and visual way to map new features here
Consider new system integrations

When scaling, pressure increases on a digital program to integrate with new or different electronic systems, especially when transitioning a program over to a government that may have other national systems — unique ID systems, supply chain management systems, health information systems — whose data may be linked or related to your solution’s data in some way. In addition, you may need to work with different partners in different locations, including mobile network operators (MNOs), aggregators or other service providers, and integrate with their systems. Governments or other partners may use specific standards for data exchange, and your application programming interface (API) and its documentation may need to be updated. Gathering API documentation from other system providers and creating a visual data-flow diagram that shows information-system dependencies is one initial step many organizations perform in the design phase.

Consider updating your data model to accommodate scale

A program at scale needs to accommodate all of the data elements that might be necessary in the various situations in which the solution may be used. It’s worth spending time in the design phase to make sure your current data model has all the necessary data elements and metadata required. If you’re scaling a solution for national government, you may need to accommodate different local government structures, hierarchies and reporting needs in different states.

For example, two programs in the same country may use different spellings of location names for the same location, and may even have different district counts, and you may need to integrate with both. Similarly, your solution may need more drop-down fields and validation rules to ensure that data entered is valid.

The more free text fields you have, the more challenging it will be to maintain data quality at scale. At scale, you are also likely to need to store more metadata about how, when and by whom records were created, so you can track user activity or troubleshoot issues.
Consider external factors likely to impact solution design

When we first launched Kilkari, our IVR-based, mobile health education service for new and expecting mothers in Bihar, India, in 2013, it was designed as a cross operator service. Anyone in the state could subscribe to Kilkari by dialing short codes, and pay for the service using their mobile phone credit, thanks to our partnerships with the six largest MNOs in India, and an Indian aggregator.

To minimize MNO network costs, and thus the tariff billed to the end user, we agreed on a decentralized solution design. Local outbound calls to subscribers were made using the IVR infrastructure owned by each MNO in the state. This local infrastructure communicated with our aggregator’s centralized IVR platform in another state to access subscriber records, and to record usage habits.

We opened the same subscription short code across all six networks in the state, configuring their billing systems to deduct a minimal subscription fee from each subscriber each week. This decentralized, cross-operator, short-code, end-user billing approach took a huge amount of effort to set up. Configuring and testing short codes, tariffs, and subscription billing across six MNOs’ network switching centers in Bihar took a considerable amount of time, and maintaining the system was complex.

When the Government of India decided to adopt and nationally scale Kilkari, we collaboratively revised our solution design. In many ways, we made it simpler:

- Firstly, the government decided to cover call costs and make the service free, which meant we no longer had to integrate with six MNOs’ billing systems.
- Secondly, the government asked us to automatically subscribe the millions of women already registered with unique mobile numbers in its Health Management Information System (HMIS).
- Thirdly, the complexity of configuring cross-operator subscription short codes was eliminated, and a much simpler toll-free, long-code (a normal 10-digit mobile number) approach could be implemented.

MNOs already have pre-existing interconnect agreements for routing calls to each other’s long codes in India (anyone from any network can call a long code on another network), so the government only had to procure one MNO partner to enable universal access. Finally, the government could negotiate much lower rates for long distance calls because millions of calls were projected.
Consider how new regulations, policies or laws might impact your solution design

Before finalizing your solution design, review whether existing regulatory guidelines or legislation has changed or whether you need to comply with different rules in a new geography. Telecommunications, IT and privacy legislation can have a profound effect on solution design.

For example, new privacy laws may require you to take consent differently, such as integrating with independent third-party systems, which could be expensive and time consuming. Or you may currently work in a country that allows personal data to be hosted on a foreign cloud, such as Amazon data servers in the United States or Singapore, but are planning to scale to a country where legally you must host personal data on local machines. This could mean changing your entire infrastructure strategy.

FOR MORE INFORMATION SEE:

Legal, Policy and Regulatory: all steps

---

Before finalizing your solution design, review whether existing regulatory guidelines or legislation has changed or if you need to comply with different rules in a new geography.

Consider streamlining your reporting requirements

The organizations interviewed for this guide highlighted reporting requirements as a key consideration often overlooked during time-pressured scale ups. Scaling to new locations usually brings new reporting requirements and a need to aggregate data from multiple databases. Reports may need to be generated from several databases potentially owned and managed by different stakeholders. New partners may also bring reporting requirements to support their business processes: MNOs may require validation of billable call minutes and SMS or data usage every month, or governments may want performance-monitoring reports for local staff delivered in different formats.
Other examples

Watch Peter Benjamin of HealthEnabled speak about the need for a solution to be almost boring to reach scale without drastically increasing tech-support time and forcing developers to work through the night.

How to

- Compare data protection laws around the world [here](#)
- Read about some simple ways to improve data security [here](#)
- Read more on techniques to select appropriate reporting indicators [here](#)

Changes to reporting systems may be minimal when scaling an existing solution in the same geography, but when extending to a new geography, it’s critical to review and streamline existing reports and indicators. When you are several years into a digital program, it’s normal for dozens of hand-crafted queries to exist, built over the years to satisfy the custom reporting needs of different stakeholders. Conducting an audit of existing reports and validating what is still useful to stakeholders can help declutter your reporting system and improve its performance.

No matter how you scale, your reporting infrastructure will likely need to handle increased user load, and enable more flexibility for report customization. Strategies to address this include hosting reporting servers separate from application or database servers, so performance of the solution itself is not affected by increased report generation, and using third-party business intelligence reporting software instead of hard-coding reports.
Agree to target features for a minimum viable product
Having completed the previous steps, you should now be able to determine whether your existing solution needs to be refined or rebuilt from scratch. Implementing organizations have found it helpful to agree a minimum set of target features with developers and stakeholders. These target features must be delivered before a minimum viable product can be launched. Depending on how experienced your team is in managing feature requests and conducting triage overall, as well as the volume of new requests needed, this activity could be done quickly, or take a long time.

Work with stakeholders to get buy-in
Include stakeholders, both old and new, in discussions to ensure that your redesign meets their needs. Hold workshops with stakeholders to validate feature requirements or vote on feature priorities to secure their buy-in. This process requires significant investment in documenting and showcasing the new solution. Your team must be able to clearly demonstrate how the solution addresses key user stories through nontechnical requirements documentation, visual mockups or prototypes. Product owners or managers are responsible for understanding all user persona needs, translating those needs into functional requirements and prioritizing those requirements with development teams. Including your product owners and development team members in the documentation process will speed up showcasing and buy-in.

Product owners or managers are responsible for understanding all user persona needs, translating those needs into functional requirements and prioritizing those requirements with development teams.
How to

- Find interactive techniques for prioritizing features with stakeholders here, here and here
- Read more about feature prioritization strategies here

When scaling to a new geography, budget additional time and effort for feature prioritization

When scaling your solution to new geographies, the feature prioritization process doesn’t change but the resulting priority features might. It may take substantially longer to develop features for scale up in a new geography than scaling in the same geography, which typically requires smaller tweaks and verification of design choices. And it will require much more time up front to set the solution context for stakeholders who may be unfamiliar with your program. By the end of this process, you should have a prioritized list of feature requests that programmers can use in the development phase.
Evaluate standard systems with professional support

In the pilot phase, you may have built your own digital solution or partnered with a small technology solution provider to build it or used their existing platform. If you’re now planning to scale, potentially in partnership with a national government or private-sector partner, it is time to decide whether you’re going to enhance your existing solution, rebuild it entirely or use third party software that is already being used at scale. Your decision will be informed by your solution design, the number of users you expect, the frequency and volume of their usage, how much funding or investment you have and your time to launch.

If you’re an NGO making a big commitment to a donor or government to deliver sustainable impact at scale, it’s worth researching the open source and proprietary software markets to identify whether the software solutions you need already exist. For example, there are numerous content management systems, profile management systems, customer relationship management systems, campaign managers, management information systems, business analysis tools, dashboards, and SMS and IVR systems that are readily available, have already been deployed at scale and could be configured to meet your needs.

Use requests for information or proposals for a structured, in-depth comparison of software options

Once you’ve come up with a long list of the software products that might work for your solution, consider sending out a request for information (RFI) to the companies that built them, resell them or offer them as fully managed services. You will then have structured information to compare and rank different organizations’ software solutions, including your own, against your requirements. Using software, proprietary or open source, that has already been tested at scale might be a less risky option than trying to expand custom-built software which has not yet been used at scale.
In practice | BBC Media Action

How we used RFIs and RFPs to create a world class system

When we first launched our mobile health education services in Bihar, India, in 2012 and 2013, they were powered by an early version of an open-source software solution developed by an NGO in the United States. This back-end system was integrated with a proprietary interactive voice response system (IVRS), owned by an Indian aggregator, and with an open source Management Information System (MIS), configured by our NGO partner.

We procured the IVRS on a fully-managed service basis, i.e. the IVRS was supported and hosted by the aggregator in their data center.

When the Government of India decided in 2014 to scale two of our services nationally, we began working with the Ministry of Health and Family Welfare (MoHFW) and our NGO partner to scale the end-to-end solution to handle calls pan India.

We decided to issue a Request For Information (RFI) for the IVRS, which detailed our requirements including updating our IVR applications, providing a configurable CMS for audio content, logical call handling capability, multi-directional connectivity, APIs to integrate with our open-source backend solution, procurement of hosting infrastructure, and provision of AMCs and L1 and L2 support in the government’s data center.

We issued the RFI because our software licensing requirements and hosting approach had changed, and we wanted information on new software solutions that might meet our new requirements and help us refine our solution design, and to see whether any strong competitors to our existing aggregator had emerged in the market.

Our NGO partner also decided to issue an RFI, not to assess whether there were any robust alternatives to its software, but to identify a system integration (SI) in India that could configure, deploy, integrate and test the latest version of the software, as well as support it in the government’s data center after launch. There was concern in the stakeholder consortium about going with the latest version of the back-end system, because the software was still being built and hadn’t been beta tested yet, but a decision was made to proceed.

After analyzing the results of the RFI, we finalized our solution design and requirements and released an RFP for the IVRS to a shortlist of commercial vendors, asking for quotes. The RFP included our technical and functional requirements, a statement of compliance, our licensing requirements, the Bill of Materials for the infrastructure we needed procured, our support requirements, the SLAs we needed the vendor to agree to and a draft agreement with our legal terms.
Once you’ve established that a short list of organizations and software products meets your requirements, considering sending out a request for proposal (RFP) to the shortlisted organizations to get estimates, quotes and timelines. You may need to send out separate RFPs for different aspects of your solution. For example, you might want to procure an IVR system from one vendor and a profile management system from another. Depending on your approach to hosting and infrastructure, you may need to go through an RFP process to procure infrastructure too.

**Reevaluate your hosting strategy**

Using cloud providers can drastically lower capital costs and increase flexibility for scaling your infrastructure up or down as needed. Cloud hosting also reduces the need for local resources to maintain local servers. The cloud-services industry is advancing at an extremely rapid pace, so even if your system is currently hosted locally on your own hardware, this is a good time to research available cloud services and consider changing your hosting model. Ensure that the privacy and data protection laws in your country allow it, as many countries do not allow personal data to be hosted on a cloud server located outside of the country.

If local hosting is required, new third-party software and hardware needs to be specified and procured by you, or procured by the government if you’re transitioning to government. Be sure to research third-party software and hardware suppliers in the same markets where you will deploy your solution. Local distributors may have already imported equipment, cleared customs and paid duties. All third-party hardware and software licenses should come with annual maintenance contracts, which you need to make sure comply with your service level agreement (SLA) requirements.

**FOR MORE INFORMATION SEE:**

Legal, Policy and Regulatory: **STEP 1. ASSESS THE IMPACT OF EVOLVING LAWS, POLICIES AND REGULATION ON COMPLIANCE**

**Assess whether you or your technology partner has the staffing in place to deliver**

As part of the process of comparing your existing solution to other software products, it’s worth considering whether you or your technical partner has the staffing and processes in place to deliver at scale. Small organizations or teams involved in building software solutions may outsource development to another organization, which may not be engaging with the product every day. Or they may bring in a few star developers who cover the whole software development life cycle, including everything from new feature requests to testing to writing deployment manuals.

These approaches often struggle to scale, resulting in poorly written code — much of which has been hard coded or no longer reflects the needs of the solution — or overburdened staff who eventually quit. If this is not addressed in your planning process, scaling can result in a buggy solution that alienates users and takes months of expensive work to stabilize.
In practice | BBC Media Action

How we used RFIs and RFPs to create a world class system (continued from page 137)

In response to our RFP, several vendors demonstrated that they had carrier grade, enterprise level IVR systems that had been deployed by multiple MNOs at scale around the world, but ultimately only one was willing to meet our licensing requirements, and was selected.

The delivery of the IVRS, including integration with the government’s MNO connectivity and procurement and deployment of infrastructure, went according to plan. Unfortunately, the scale up of the back-end solution proved complex and much more time consuming than anticipated, partly because the SI contracted by our NGO partner could not master the complex code-base. The services were successfully launched, and the back-end solution has proved a powerful and robust tool — but challenging to configure and support without a dedicated team of senior engineers.

Other examples

> Watch Ravi Kiran from IMImobile talk about how a technology partner might assess the infrastructure requirements for a solution. See an example of the sizing sheet he mentions here.

HOW TO

> See an example form to evaluate bidders for technical RFPs here.

FOR MORE INFORMATION SEE:

> Human Capacity: STEP 1. REASSESS YOUR HUMAN RESOURCE STRATEGY
## Typical components of an RFP

<table>
<thead>
<tr>
<th>Module</th>
<th>Content to include in an RFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business model and commercial terms</td>
<td>Include a request for a detailed budget or commercial terms for the work, based on effort estimates. Also describe the business model you need. For example, if you’re seeking an aggregator, you may want to work with them on a revenue-share basis.</td>
</tr>
<tr>
<td>Legal, policy and regulatory</td>
<td>State your licensing terms. For example, you will need a license that allows for the number of users you expect to acquire and retain. And you may need it to be transferrable to the government. Include the SLA that you need the vendor to sign, guaranteeing a certain level of system availability and response times for resolving bugs or other technical issues according to level of severity, and an escalation matrix. Include a draft of the contract that you need the vendor to sign. Sending this out with the RFP and requiring vendors to mark up the draft if they can’t comply with any of your legal terms can save you a lot of time later. You may find the perfect piece of software but the legal terms may not be acceptable to your organization.</td>
</tr>
<tr>
<td>Solution design</td>
<td>Include a statement of compliance, listing all your feature requirements, where the vendor will indicate whether the feature already exists and is easily configured, is partially built, or needs to be built from scratch (and how much effort this would take).</td>
</tr>
</tbody>
</table>

For more information see these modules:

- Business Model
- Legal, Policy and Regulatory
- Partner Relationships
- Human Capacity
- Roll Out
- Roll Out
<table>
<thead>
<tr>
<th>Module</th>
<th>Content to include in an RFP</th>
</tr>
</thead>
</table>
| **Roll out**           | Request a project plan for testing, integrating, configuring and deploying your solution and then fixing problems discovered post-deployment after the system is in regular use.  
                        | Describe the support that you will require after launch. You may require an annual maintenance contract for the software, which typically costs nine to 11 percent of the value of the software license.  
                        | Ask what support the vendor will provide for ongoing operations, maintenance and monitoring once the solution has been rolled out.  
                        | Ask how and with what frequency the vendor will provide regular updates for the solution. Will the vendor push updates or will you need to pull them? Will the vendor update the solution when updates are released for external software dependencies?                                                                                       |
| **Human capacity**     | Request a staffing matrix, short bios for the key staff that will deliver the work, and an escalation matrix.  
                        | Vendors should provide you with an account manager (usually not dedicated) to act as your single point of contact. They should also budget for an experienced technical project manager or Scrum Master to manage the work of the engineering team. Other staff may include a business analyst, a technical solution architect, and test and quality assurance (QA) teams. It is best practice for the QA team to be separate from the development team. |
| **Partner relationships** | Ask for references for previous clients that the organization has worked with in the past and examples of similar work successfully delivered.                                                                                                                                                                                                                                                                             |
Reinforce your development team

As previously described, many organizations originally outsource development to another organization or brought in a few star developers. These approaches often struggle to scale. If this staffing policy is not addressed in your planning process, scaling often results in high drop-out rates when customers stop using the existing or new solution because bugs are not fixed fast enough, or the feature set does not meet their needs.

Some organizations suggest bringing on new developers, who are specialists in a single technology. They then lead the development of specific features, or manage functional areas, such application performance tuning. This has proved more effective than tasking existing developers with work on potentially unfamiliar technologies.

FOR MORE INFORMATION SEE:

▶ Human Capacity: STEP 1. REASSESS YOUR HUMAN RESOURCE STRATEGY

Perform a code and backlog review

Replication usually requires improvements in code to remove performance bottlenecks. Diversification requires code changes to implement new features and add flexibility and configurability to systems. The software development team should review the existing code base to identify and recommend technical fixes and improvements to make the code base more reliable, more efficient and easier to manage. Most development teams will already know many of the problems and track them in a backlog of fixes. These fixes should be added to, and prioritized with, the previously identified feature changes and improvements.
In practice | BeeHyv

Watch Ajai Adusumilli from BeeHyv talk about the importance of load testing your solution and decentralizing system architecture components to prevent performance bottlenecks.

How to

- Read more about Agile methodologies [here](#)
- Learn how to create detailed software development LOE estimates [here](#)
- Find popular tools to create, prioritize and track development tasks [here], [here], [here] and [here](#)

Replication usually requires improvements in code to remove performance bottlenecks. Diversification requires code changes to implement new features and add flexibility and configurability to systems.
**Revise your development roadmap or release plan**

The software development team should then review the complete set of changes identified, including new features, changes to existing features and fixes to address identified bottlenecks. For each change, the team will estimate the amount of time and effort required to implement it. These level of effort (LOE) estimates can then be used to prioritize and organize features into a development roadmap or release plan.

It is not unusual to be faced with more work than is feasible to perform within the project timeline and budget. It is critical that code changes, which are engineering tasks, be prioritized with LOE estimates taken into account. Sometimes lower priority but quicker or easier features will be prioritized. And it is almost always the case that features will be dropped from the list to bring the overall estimated effort in line with project timelines and budget.

With a reprioritized and estimated set of code changes in hand, the development team can create a release plan — a schedule of when changes are expected to be completed and when these changes, in the form of a release, can be deployed for testing, staging and production use. Once completed, the release schedule or roadmap is a powerful tool for communicating system capability and timelines to stakeholders.

**Build, test, modify and repeat**

When the planning is done, the engineering team will begin implementing the changes. It is critical that the non-technical program team remains involved in this phase. Best practice is to use one of the Agile project management methodologies. There are various codified Agile methodologies, but all of them share the same core philosophy: build, test and modify in short cycles.

A key strategy for both types of scale is to build rapid software prototyping and usability testing into the software development life cycle. It is more important when replicating to make incremental changes and test them thoroughly before releasing them. When diversifying, it is important to be able to quickly test several very different versions of the solution to determine which will have the highest acceptance.

Technical infrastructure must be set up to support this process, specifically testing and staging environments where pre-release systems can run without impacting the production environment. Staff must plan for and commit significant time to reviewing test systems and providing feedback to developers, which the developers subsequently incorporate in the next development effort.
It is not unusual to be faced with more work than is feasible to perform within the project timeline and budget. It is critical that code changes, which are engineering tasks, be prioritized with LOE estimates taken into account.
Deploy and migrate (if needed)

Migration of data from an old to a new version of a system is a significant challenge that is often underestimated. Changes to a solution often include changes to its underlying stored data, which might require thousands or millions of previous records to be updated to fit into the new database structure. This may require significant data cleansing. System downtime may also be required to switch over from one hosted infrastructure to another.

Whether to migrate is a decision that can only be made by comparing the total cost of old and new infrastructure. Developing a migration plan that identifies when specific locations will switch over to the new system is key. Optimally, switchover is timed during periods of low usage. The migration plan should also include a communication plan to notify stakeholders of downtime, a testing plan to validate that the new system is up and functional before making it live, and a fallback plan in event of failure. A data migration plan that specifies which data should be moved from the old system to the new, and which data should be left behind when records are migrated is also key.

FOR MORE INFORMATION SEE:

Business Model: STEP 2. ESTIMATE THE TOTAL COST OF OWNING YOUR DIGITAL SOLUTION

Find the right time and frequency to release updates

Keeping your solution current with the latest updates and security patches is critical. Incidents of widespread hacking, malware and ransomware attacks have increased public attention on the consequences of failing to keep solutions secure. Many organizations do not plan for this and neglect to patch the software platforms used in their solution until they run into a problem. Encouraging your developers to stay current with security updates and allocating time for routine software patching and maintenance will help you get ahead of potential issues. If your
Other examples

- One of the biggest challenges encountered by Echo Mobile’s program which uses sensors to track matatu (shared taxi) driving behavior in Kenya to improve productivity and safety, has been the operational costs of growing user support needs. Although users were submitting daily feedback via toll-free SMS, the Echo Mobile team often needed to make follow-up calls or in-person visits to users if the feedback submitted was unclear or inaccurate. The previously allocated budget for user support did not adequately cover the time and costs required by the team to make these important calls and visits. This meant that Echo Mobile had to reallocate budget from a different part of the program to cover the overrun in user support costs. Read more about the program here.

- D-tree has many programs that distribute phones and monthly data or airtime bundles to community health workers (CHWs). They also face challenges with CHWs losing or breaking their phones and insufficient funding to replace them. They learned that CHWs who received a flat monthly stipend, regardless of how they use the digital tool, were not incentivized to keep their phones safe or use their mobile data only for work purposes. To address this, they withheld a small percentage of CHW stipends (approximately USD $1.50 per CHW in each group) and added it to a funding pool used to replace lost phones. At the end of each year, any remaining money in the pool was split evenly among the group, providing a positive incentive for CHWs to keep their phones safe.

How to

- Read more about data cleaning here
- Read more about the optimal frequency for updating mobile apps here
- Learn how to calculate the costs of migrating to the cloud here
- Read more about strategies to migrate to the cloud here and here
solution is hosted by an external provider, they may be responsible for patching servers and software dependencies. Ensure that this is fully described in your SLAs.

In addition to patching the core software that your solution depends on, you will also need to apply or send out patches for your own solution. If your solution is entirely web-based or call-in based, you may only need to apply patches to software on your central servers. If your solution has a mobile client or app, you will need to push updates to your users’ mobile devices or allow them to download the updates on their own. Distributing updates to existing users may cause them to incur charges for downloads. You may need to build this consideration into your release schedule and ensure updates are not made too frequently. Ensuring that support staff are trained in new features, and expecting an increased workload after substantial updates, will help make roll outs of software updates more manageable.

FOR MORE INFORMATION SEE:

Legal, Policy and Regulatory: STEP 6. REVIEW AND UPDATE SERVICE LEVEL AGREEMENTS (SLAS)

Plan for ongoing operational costs

Operational costs are often overlooked in solution design. The costs of supporting users and maintaining a solution at scale can be significantly higher than the cost of developing an innovative solution. Many organizations do not fully understand operational costs until they start dealing with technical issues, at which point the costs may be severe. If users do not receive immediate support when they encounter issues, they may not trust nor want to use the solution again. Once trust is lost, it may be impossible to get back, making it even more critical that support structures are financially planned for and put in place before the solution is rolled out.

FOR MORE INFORMATION SEE:

Business Model: STEP 2. ESTIMATE THE TOTAL COST OF OWNING YOUR DIGITAL SOLUTION

Roll Out: STEP 5. DEVELOP SUSTAINABLE, COST-EFFECTIVE DISTRIBUTION CHANNELS AT SCALE
The costs of supporting users and maintaining a solution at scale can be significantly higher than the cost of developing an innovative solution.
RESRESOURCE ROLLUP

Who do you need?

Expanding your digital development program requires bringing in new skills and augmenting your current teams. The table below indicates the types of human resources you will need to complete the steps in this module. Your partners’ support with technology design, development, deployment, hosting and maintenance will be integral to sustaining your program at scale.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Resource type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze, design and prioritize features</td>
<td>Business Analyst, User Interface/User Experience Designer</td>
</tr>
<tr>
<td>Development and maintenance</td>
<td>Technical Lead, Software Developers</td>
</tr>
<tr>
<td>Manage technical partners and deployment</td>
<td>Project Manager / SCRUM Manager</td>
</tr>
<tr>
<td>Data migration</td>
<td>Database Administrator</td>
</tr>
</tbody>
</table>

Pro tips

- **Choose proven technologies.** No matter how skilled your software development team, adopting and customizing an existing technology platform that has been proven to work at scale can save you time, effort and money in the long run.

- **Focus on support over features.** Maintaining and operating a software system is a complex effort. When selecting software components, focusing on the availability and cost of ongoing support to address problems and extend the system’s capabilities will make your project more resilient to both technical issues and changing needs.

- **Be modular.** The more you rely on self-contained software modules and services, the easier it will be to troubleshoot issues, replace old or malfunctioning modules with improved ones, and survive when key developers leave. Avoid monolithic software architecture.

- **Choose flexible hosting.** While international cloud hosting is not a legally viable option in every country today, its infrastructure-as-a-service model offers immense benefits to programs that need flexibility to scale up and down as needed. Many governments are building in-country cloud hosting services to reap these rewards. Designing a solution that can switch to cloud hosting is designing for the future.
<table>
<thead>
<tr>
<th>Key step</th>
<th>Referenced Resources</th>
</tr>
</thead>
</table>
| **1. USE A HUMAN-CENTERED DESIGN APPROACH TO REASSESS YOUR USER REQUIREMENTS** | • Guide: User Journeys – The Beginner’s Guide  
• Article: When and How to Create Customer Journey Maps  
• Guide: The 5 Best Guides for Creating Customer Personas  
• Example: User Personas  
• Guide: The Field Guide to Human-Centered Design |
| **2. CONSIDER EXTERNAL FACTORS THAT MAY CHANGE YOUR EXISTING SOLUTION DESIGN** | • Tool: Compare Data Protection Laws around the World  
• Article: 5 Simple Ways to Improve Your Data Security Today  
• Guide: Monitoring and Evaluating Digital Health Interventions (Part 2c: Setting the Stage: Selecting Indicators for Digital Health Interventions) |
| **3. PRIORITIZE NEW FEATURES AND SET TARGETS**                          | • Tool: Innovation Games: Buy a Feature  
• Tool: Innovation Games: 20/20 Vision  
• Toolkit: 6 Backlog Prioritization Techniques  
• Article: Agile Best Practice: Prioritized Requirements |
| **4. PLAN YOUR TECHNICAL DEVELOPMENT FOR SUSTAINABILITY AT SCALE**       | • Example: Inbound IVR Sizing  
• Example: Statement of Compliance  
• Example: RFP Evaluation Form |
| **5. ENHANCE YOUR DEVELOPMENT PROCESSES IF YOU DECIDE TO BUILD YOUR SOLUTION** | • Article: Agile Software Development  
• Tool: A Simple Project Effort Estimation Utility  
• Tool: PivotalTracker  
• Tool: GitLab  
• Tool: Asana  
• Tool: Visual Studio Team Services |
<table>
<thead>
<tr>
<th>Key step</th>
<th>Referenced Resources</th>
</tr>
</thead>
</table>
| 6. DEPLOY, MIGRATE AND MAINTAIN YOUR SCALED SOLUTION | • Article: Data Cleaning, Management, and Tagging: The Best Practices  
• Article: How Often Should You Update Your App?  
• Article: How to Calculate the True Cost of Migrating to the Cloud  
• Article: 6 Strategies for Migrating Applications to the Cloud  
• Guide: Enterprise Cloud Strategy (Moving IT to the Cloud) |
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile methodology</td>
<td>An approach to project management utilized in software development. It uses incremental, iterative work sequences, commonly known as sprints.</td>
<td></td>
</tr>
<tr>
<td>Aggregator Source</td>
<td>An organization that acts as a middleman between application and content providers, and mobile carriers. Provides message traffic throughput to multiple wireless operators or other aggregators; provides mobile initiative campaign oversight, and administration, as well as billing services.</td>
<td></td>
</tr>
<tr>
<td>Annual maintenance contract Source</td>
<td>A written document that sets forth the terms of an agreement between a client and a maintenance service provider. The overall purpose for a maintenance contract is to have consistent fees and regular availability of the service provider at a savings over random or emergency calls.</td>
<td></td>
</tr>
<tr>
<td>Application programming interface (API) Source</td>
<td>A set of protocols, routines, functions or commands that programmers use to develop software or facilitate interaction between distinct systems.</td>
<td></td>
</tr>
<tr>
<td>Business intelligence (BI) Source</td>
<td>Any information that pertains to the history, current status or future projections of an organization. BI software enables users to obtain enterprise-wide information more easily, letting users slice and dice the information from their organization’s numerous databases without having to wait for their IT departments to develop complex queries.</td>
<td></td>
</tr>
<tr>
<td>Carrier grade Source</td>
<td>A system or a hardware or software component that is extremely reliable, well tested and proven in its capabilities.</td>
<td></td>
</tr>
<tr>
<td>Cross-operator</td>
<td>A service that is made available by multiple mobile network operators (MNO), typically in an identical fashion. For example, dialing the number “911” will call emergency responders in many countries in the world, regardless of the MNO being used to make the call.</td>
<td></td>
</tr>
<tr>
<td>Data-flow diagram Source</td>
<td>A graphical representation of the “flow” of data through an information system, modelling its process aspects.</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Drop-down fields</td>
<td>A data field that allows the user to choose one value from a list.</td>
<td></td>
</tr>
<tr>
<td>Health management information system (HMIS)</td>
<td>An information system specially designed to assist in the management and planning of health programs, as opposed to delivery of care.</td>
<td></td>
</tr>
<tr>
<td>Interactive voice response (IVR)</td>
<td>An automated telephony system that interacts with callers by gathering information and routing calls to the appropriate recipient.</td>
<td></td>
</tr>
<tr>
<td>Interconnect agreement</td>
<td>A business contract between telecommunication organizations for interconnecting their networks and exchanging telecommunication traffic.</td>
<td></td>
</tr>
<tr>
<td>Level 1 support</td>
<td>Basic help desk resolution with lower-level technical personnel trained to support simple customer issues, such as solving usage issues and fulfilling service desk requests that need IT involvement. Also called Tier 1 support.</td>
<td></td>
</tr>
<tr>
<td>Level 2 support</td>
<td>In-depth technical support from experienced and knowledgeable technicians, but not necessarily engineers or programmers, for problems that cannot be handed by Level 1 support. Also called Tier 2 support.</td>
<td></td>
</tr>
<tr>
<td>Level of effort (LOE)</td>
<td>The most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete, uncertain and noisy input.</td>
<td></td>
</tr>
<tr>
<td>Long code</td>
<td>A normal 10-digit mobile number used for two-way communication.</td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>Data that describes other data. For example, data dictionaries and repositories provide information about the data elements in a database.</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Minimum viable product</td>
<td>The most pared down version of a product that can still be released.</td>
<td></td>
</tr>
<tr>
<td>Product owners</td>
<td>The product owner represents the product’s stakeholders and the voice of the customer; and is accountable for ensuring that the team delivers value to the business. The product owner defines the product in customer-centric terms (typically user stories), adds them to the product backlog, and prioritizes them based on importance and dependencies.</td>
<td></td>
</tr>
<tr>
<td>Queries</td>
<td>Searches performed on a database. In addition to obtaining lists of records that match the search criteria, queries to a database allow for counting items and summing amounts.</td>
<td></td>
</tr>
<tr>
<td>Request for information</td>
<td>A process that uses a standardized format for collecting information from possible vendors or suppliers</td>
<td></td>
</tr>
<tr>
<td>(RFI)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request for proposal</td>
<td>A document used to solicit detailed proposals or bids from vendors or suppliers</td>
<td></td>
</tr>
<tr>
<td>(RFP) or Request for</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quote (RFQ)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service level agreements</td>
<td>A contract between a service provider (either internal or external) and an end user that defines the level of service expected from the service provider. SLAs are output-based specifically defining what the customer will receive. SLAs do not define how the service is provided or delivered. The metrics that define levels of service should aim to guarantee: a description of the service being provided, reliability, responsiveness, procedure for reporting problems, monitoring and reporting service level, consequences for not meeting service obligations, and escape clauses or constraints.</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Short code</strong></td>
<td>A numeric code assigned to a commercial organization for SMS text messaging. Users send a message to a short code to receive a prepared (not interactive) response.</td>
<td></td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Software development life cycle</strong></td>
<td>A framework defining tasks performed at each step in the software development process consisting of a detailed plan which describes how to develop, maintain and replace software. The life cycle defines a methodology for improving the quality of software and the overall development process.</td>
<td></td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Unstructured Supplementary Service Data (USSD)</strong></td>
<td>Sometimes referred to as quick codes or feature codes, USSD is a protocol used by cellular telephones to communicate with the service provider’s computers. Unlike SMS text messages, USSD messages create a real-time connection which remains open, allowing for a two-way exchange of data. This makes USSD more responsive than services that use SMS.</td>
<td></td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Visual mockups</strong></td>
<td>Rules built into a software’s user interface that validate if data entered into a data field is acceptable and prevent erroneous data from being entered.</td>
<td></td>
</tr>
<tr>
<td><strong>Visual mockups</strong></td>
<td>A visual representation of how a digital solution’s user interface will look once it has been developed. Mockups can be simple sketches on paper or more detailed visualizations created using visual design software.</td>
<td></td>
</tr>
</tbody>
</table>