WA2325 Solution Architecture (SA) Practitioner's Guide (Extended)

Student Labs

Web Age Solutions Inc.

Table of Contents

Lab 1 - Defining the Problem

Through the labs in this course, we will develop the solution architecture for an application to solve a hypothetical business problem. The purpose of the exercises is to practice what we discuss in lectures; we will not go into the level of detail that would be required for a complete solution architecture.

Labs follow an order, more or less the sequence you would follow when developing a solution architecture for a new system. If you miss a lab or don't complete everything in a lab, don't worry - you don't need a completed lab to move on to the next lab, and you can fill in any gaps with your imagination where you need to.

There are some differences between what we do in the labs and what we do in the real world. For the purpose of the course, we go through labs sequentially. In the real world, developing a solution architecture is iterative we work on one part, move on, learn more, and return to refine or correct, often several times. Developing a solution architecture is seldom a straightforward, predictable process.

Developing a solution architecture for an enhancement is also different from developing a solution architecture for a new system. With an enhancement, we are changing a system that already exists. The solution architecture activities may involve discovering the architecture, or relevant parts of the architecture, of the as-is solution. The creative effort is about the changes we need to introduce to have the desired result. This may sound easier, because we are likely changing "only" a small part of the whole, but often it's actually harder; we have to figure out how to change the solution's behaviour or characteristics without breaking what works. There are more constraints, and often more trade-offs to consider.

Keep these differences from the real world in mind when doing the labs. The real world is richer, more challenging, and hopefully more rewarding than the simplified environment of the labs!

Part 1 - Get organized

Divide into groups of between three and six people.

After each lab, you will share your work and insights with the class; we suggest selecting one person to be the speaker for the group and have this role rotate as you go through each lab.

Part 2 - Select your system

We will develop the solution architecture for a system through a series of labs. You have two options for the system that you work on

- Use a provided example
- Come up with a system of your own

If everyone on your team is from one organization, and you have a problem in mind that is suitable for the labs, we encourage you to come up with your own system to study. Your familiarity with the problem may make it more relevant and you will likely learn more about your organization from each other.

The provided example is designed to have all the elements needed to be relevant for all the labs. We also provide sample solutions to each lab, so you can follow the reasoning from the problem to the questions raised in each lab to the resulting artifacts that make up the solution architecture.

The provided example is described in Lab 9. It involves a paint manufacturer moving into the consumer market, and an application that it intends to build to support its resellers.

If you wish to use a problem of your own, make sure that it is suitable for the labs. You want a problem that is large enough that you will need to think about multiple aspects of the solution

- There should be multiple stakeholders interested in the solution
- There should be a business issue the solution addresses, and a business context the solution fits into
- You should have some quality of service concerns, such as availability or extensibility
- The solution should involve data in a significant way
- You should be able to think about where and how the solution would be deployed

Solution architecture projects can emphasize different things; some are data-centric, some revolve around integration, while some are focused primarily on infrastructure. For the purpose of the labs, we suggest selecting a problem that revolves around an application, preferably the development of a new application. This is not to suggest any bias in what solution architects should be doing, it is just guiding you towards a problem that will have something of interest for every lab. If you select a problem that is not application-centric, you may find that some labs do not fully apply.

Discuss the problem you are thinking about with your instructor, who may have hints or suggestions.

Part 3 - Preparation for using the provided example

If your group has chosen to use the provided example,

- Individually, read Lab 9
- As a group, discuss your understanding of the problem

Your instructor is available if you have any questions or would like clarification.

Part 4 - Preparation for using a system of your own

If your group has chosen to use a system of your own

- As a group, discuss the problem that you would like to address
- Write down, at a high level, key aspects of the problem; make sure you capture
  - The business problem (what is the motivation)
  - How the business problem will be addressed (what the solution is, in business terms - stick to a high level description but call out the key features of the solution if applicable)

Use the provided example as a guide to what you may want to think about; it is not necessary to try to write down a problem with the level of detail used in the example!

Part 5 - Wrap-up

Once you are satisfied that you understand the problem you are going to address, you are done. If you have chosen to make up your own problem to address, you will be asked to briefly describe the problem at the take-up of the next lab.

Lab 2 - Defining the Boundary

This lab explores some ways that solution architects discover and document the scope of a system. A context diagram shows the system in relation to its environment, and actors identify the humans and machines that interact with a system. Interactions happen at interfaces.

At this point we often start to think about what is inside the system - how it does what it does. Subsystems help us talk about major areas of functionality or behaviour.

Part 1 - Create a Context Diagram

In a context diagram, the system is represented as a "black box", having no internal structure. For the purposes of the context diagram, the "context" is the collection of things that interact with the system, both humans and other systems.

- Identify the boundary of your application and create a context diagram.
- Identify all the actors - anyone or any thing that interacts directly with the application. Place them on the diagram and label them.

Pro tip: A context diagram helps describe the scope of a system in terms of its context, but this visual representation does little to express what the system actually does. A scope statement, expressed as a list of things that are "in scope" and things that are "out of scope" is important to more clearly define what the system actually is. The scope statement is often prepared before solution architecture work starts, but a solution architect will often seek to refine the scope statement as they uncover considerations that have not been thought about yet.

Part 2 - Identify actors

The context diagram shows actors as symbols, but more detail is usually required.

- For each of the human actors in the context diagram, describe their role (when interacting with the system), their location (if that is relevant), and what functions they perform with the system (if it is not obvious).

  Also record anything else that you find interesting or unusual that could be useful in later analysis. It is not necessary to be complete or precise in compiling this list.

- For each of the machine actors in the context diagram, describe the actor (the "other" system) and the purpose of the interaction. Details of the interaction will be explored later in the lab.

Pro tip: Do not obsess about the list of human actors or its details; at this point in the development of the solution architecture, the main value of the identification of human actors is to help find things we may have missed. Reviewing the list with stakeholders, along with the context digram, often reveals new actors or details about actors that were overlooked.

Pro tip: Consider describing actors in terms of roles. A term like "Acme Staff" is quite generic and could include people that would never interact with the system. A term like "sales associate" is better. Sometimes specific roles are even better; consider a point of sale solution where sales associates sell things, but some sales associates have the authority to approve a return. In that case, it may be important to capture that "sales associate" and "returns approver" are two separate actors. It's OK that one is a title and the other is a role.

Part 3 - Identify key interfaces

Interfaces describe the mechanisms of interaction with the system. Interfaces are either human or machine; human interfaces are those things that people interact with - web pages, screens, and so on. Machine interfaces are some machine-to-machine interaction, which may be an API, a file transfer, or any other means.

- For each machine-to-machine interface, describe the functions performed by the interface, the data required to perform those functions, and anything that is knowable about the connecting system.

  If there is a standard API being used, specify that.

  If your system is accessing another system through an API, it is generally sufficient to simply specify that - for example "SalesForce Contact Lookup API". If it is available, point to a published interface specification.

(For the purpose of the lab, use your imagination about machine-to-machine interfaces; there is not time in the lab to research actual interfaces.)

Pro tip: We can ignore human interfaces at this point, unless there is something about them that is of architectural significance. For example, we may not need to describe a web user interface to show information about a company the user has looked up, but if company information were to include a real-time live update of stock quotes and a news feed, we would want to capture that fact because the real-time nature of the user interface is of architectural significance. Use your judgement whether to include a human interface or not.

Part 4 - Identify major subsystems

A subsystem is a part of the overall solution. We use subsystems to call out some part of the overall solution that is distinct in purpose, functionality, or structure from other parts of the solution. This definition is not meant to be vague; it is intended to be broad, and gives considerable flexibility in what we consider to be a subsystem.

- Identify major subsystems in the solution.

  Think about the big things the system does. Of those things, are there some of them that seem related? Those may be candidates for subsystems.

  Are there cross-cutting concerns? (Hint - think authorization, that's a common subsystem)

- Describe what each subsystem does in a paragraph or so.

You may find it useful to update the context diagram to show subsystems. When there are subsystems that are responsible for interactions with specific actors, you can show this on the diagram and start to show what parts of the solution any given actor interacts with.

Pro tip: Do not be overly concerned about what a subsystem is at this point; subsystems are almost a placeholder for further decomposition and analysis, and you will inevitably end up refactoring later. That doesn't mean subsystems are not important - subsystems aid stakeholders in understanding what capabilities the solution will have and what things we may have missed.

Part 5 - Retrospective

Assign a presenter and be prepared to share your work. Is there anything that you ran into while doing the exercises that surprised you, or something that you would not have expected? Share one or two things that your team learned from the exercise.

Lab 3 - Stakeholders, Views, and Viewpoints

Stakeholders - individuals with an interest in the outcome of a solution architecture project, are essential to the success of an initiative. Stakeholders are approvers and funders, and are also sources of information. Stakeholder opinion often either paves the way for the acceptance of a new solution, or puts up roadblocks.

In this lab, we look at identifying stakeholders, classifying them to determine how to meet their needs, and the views and viewpoints that will support their desire for information about the solution.

Part 1 - Identify stakeholders

Identify a number of stakeholders for your solution.

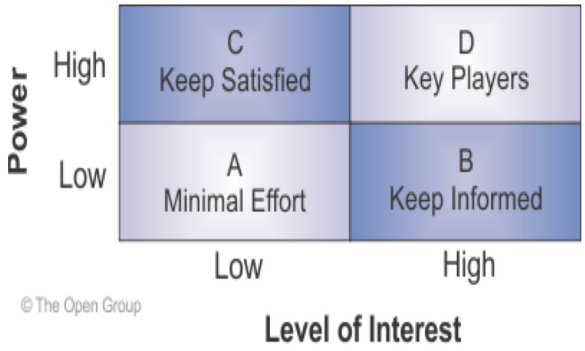
Remember that stakeholders and actors are not the same thing; there are stakeholders who do not interact directly with the system, so they are not actors.

Note that while actors are, strictly speaking, stakeholders, we don't usually go through stakeholder analysis for actors. Unless there is a need to communicate with these actor-stakeholders outside of regular design and development activities, they can probably be left out of the stakeholder list.

Part 2 - Classify stakeholders

Select no more than four stakeholders from your stakeholder list. If you have many to choose from, try to select four that are quite diverse.

Using the two-by-two matrix below, classify each of your selected stakeholders.



For each stakeholder, record what you believe their concerns would be - what are they interested in, or what do they need to know? Also record what you think would be an appropriate communications style for them, such as a regular email update, periodic face-to-face meetings, or a general update sent to multiple stakeholders.

Part 3 - Identify some views and viewpoints

Identify the viewpoint of each stakeholder and what view(s) would satisfy the viewpoint. The viewpoint will reflect what you have identified the stakeholder's concerns or interests. The view or views are artifacts that describe some aspect of the architecture, such as a context diagram or a data model.

Since we have not covered all the types of artifacts we typically encounter, focus on what needs views satisfy without getting too caught up in what specific deliverables you would use. Where you can think of specific artifacts, do list them.

Part 4 - Retrospective

Be prepared to share your work.

What did you find when going through the exercise? Did you come to a conclusion about a stakeholder or stakeholder needs that you might not have thought about before the exercise?

Lab 4 - Architecture Requirements

Architectural requirements, particularly quality fo service requirements, shape the architecture of a solution. Identifying architectural requirements, and making good decisions about trade-offs between them, is a critical activity and one that a solution architect is expected to lead.

Part 1 - Identify Quality of Service requirements

Identify at least two quality of service requirements that have architectural significance, and if possible at least one functional requirement that has architectural significance.

Part 2 - Brainstorm tactics for a quality of service requirement

Pick two architectural requirements that you feel are particularly interesting or challenging to address.

For these two architectural requirements, describe architectural options you could use for each requirement.

Depending on the nature of the requirement, there may be one obvious tactic; if so, can you come up with a few more tactics that would work, even if they seem impractical or unorthodox? (Sometimes we gravitate to the "obvious" answer, but there might be other approaches that you don't readily think of.)

For some requirements, you may find that you can employ more than one tactic at the same time. Availability, for example, is often addressed through a number of different, complimentary means.

In any case, you should be able to identify several tactics to address each requirement.

Part 3 - Assess the architectural impact of your tactics

For the two architectural requirements that you addressed, what tactics are the best options for your particular solution? Why?

Are there trade-offs that you may have to think about? Two different approaches might have different advantages, or one approach might improve one quality of service while having some other detrimental effect.

(Cost is frequently a limiting factor in determining which tactics are practical for a given situation. Cost can be dollars, but effort is another form of cost; some tactics might be impractical simply because there is not time to implement them.)

Part 4 - Retrospective

Be prepared to share your work.

What did you learn about your problem when doing this exercise? Did you find it it easy or hard to come up with architectural concerns? Does your problem have competing concerns (where improving one degrades another) and how did you decide what mattered the most?

Lab 5 - Business and Data Architecture

In this lab we explore the business domain and the data domain. Solution architecture initiatives will have varying levels of involvement with business and data; often solution architecture becomes involved after the business domain work has been done, or the business domain is neglected. Either of these can be a mistake, as a solution architect should have a deep understanding of the business role of the solution they are working on.

Data should not be neglected either; exploring the data domain is not just about architecting what is going to go into a database. In this exercise, we seek to illustrate that there is information in the business domain that will be realized in technology by the solution, in the data domain. The solution architect should keep in mind that data is not just what is stored in databases; it may be reflected in state in other systems, or it may be represented in un-traditional ways.

Part 1 - Explore the Business Domain

Identify the capabilities that the organization is changing through the project. Usually projects are undertaken to add new capabilities or improve existing capabilities.

It is rare for capabilities to be removed from an organization - remember that a capability is something the organization can do, not how it is done. A capability is implemented by business functions. While a particular function may be removed and replaced by another one (such as removal of a manual function and replacement by automation), the capability itself rarely goes away.

Identify at least one business process that the project participates in and map the process. Most solution architectures involve several business processes, but stick to one for the purposes of this exercise. Your business process should be non-trivial so that you have something to think about, and should involve some business information. (We will use the business information you identify in the next section.)

Referring to your business process, identify the most important types of business information that your solution will depend on.

Part 2 - Explore the Data Domain

Identify and model entities - conceptual and logical; if practical, model one or more of the business information items identified in the business architecture (which will help illustrate the connection between business and data architecture).

Do not be overly concerned with exactly what level of detail “conceptual” or “logical” actually is. Conceptual could be as simple as a box for each entity, with relationship lines between them. Logical could simply start adding attributes. At this point in the solution architecture, we are looking for shape and structure, so to speak, in data; we are not trying to do formal data modeling.

If you have people in your group who have the background, you may want to think about whether there are implications for a physical data model, but it is not necessary to write it down.

Part 3 - Retrospective

Discuss within your team your observations and what you learned from this exercise. Assign someone to present your conclusions and observations to the class.

Lab 6 - Application Architecture

Of all the parts of a solution architecture, the application architecture is often the largest and most involved. Application architecture is where everything comes together to create an actual application that does interesting things. That said, what an application architecture looks like varies greatly, depending on the nature of the problem and the tactics that have been chosen to address architectural requirements.

Part 1 - Decomposition

From your context diagram and subsystem diagram (Lab 2), identify major components of your solution.

The major components are usually things that could be implemented, either on their own or a few components in one package. There are lots of ways to think about this, all valid - the author tends to break an application up by technical platform first, then by capability or function next, grouping related things together. Note common references to data can be a good indication that two capabilities or functions might be closely related and possibly to be implemented in one unit.

As you think about breaking the application into parts, think about the quality fo service requirements you need to meet, and the tactics that you chose to address them (Lab 4).

Write down your first level decomposition of the application.

Part 2 - Interfaces

Refer to the interfaces that you discovered in Lab 2.

- For human interfaces, are there components in the application that will implement these function? Do you need to refactor your application? You may realize, for example, that a number fo functions sit behind an interface, so you may want to add a user interface controller component that is related to each of the functions it makes use of.
- For machine interfaces (APIs), make sure there are components in the application decomposition that handle each of the interfaces. Sometimes, you may want to add a new component as an intermediary between the internals of your application and an external machine-to-machine interface. This “proxy pattern” reduces coupling to the exterior of the application and promotes extensibility. It is things like this (additions to the architecture that are not strictly required to meet functional requirements) that show the added value of architecting a solution, rather than just going ahead and starting to build it.

Note that application interfaces (both traditional interfaces and those used solely to access data - see the next section) should be thought of as part of an enterprise integration

architecture. If an organization has standards for integration, a solution architecture is obligated to follow them (or make a case of an exception). If there are no explicit standards, a solution architect is strongly advised to follow existing practices unless there is a very good reason not to.

Part 3 - Data

You may have already started to address this, but now add data stores to your application. At this point int he process, we do not need to go into detail, but we do want to identify data stores and what entities each manages.

Data stores may be internal to the application, but they may be external as well; effectively these are interfaces, and you should treat them as such, including considerations like coupling. Knowledge of the enterprise environment and direction should help you understand what is necessary and appropriate.

You should have enough intuition about data to be able to indicate what kind of data store each is - relational database, configuration file, some NoSQL database, a cache, or whatever. These have very different behaviours and it is important to know what is being designed for in the solution architecture.

Part 4 - Retrospective

One last comment before wrapping up this exercise - A key consideration in application architecture is how the overall solution is best decomposed into smaller parts. We have pointed you towards ideas like decomposing around subsystems, around functions, and using data to help find logical groupings of components. Another consideration, which we do not address here, may be the build process. It is increasingly common to create applications using continuous delivery processes, where an application can be built, tested, packaged, and even deployed very frequently, relying on extensive automation.

It may not be obvious, but the way you decompose an application into components, and the dependencies between those components, can affect whether an application is easy or hard to build with continuous delivery practices. We feel that solution architects who are working in a continuous delivery environment should have a basic understanding of how the build tooling and delivery automation mechanisms work, and understand basic best practices for continuous integration and continuous delivery, so that they can at worst avoid getting in the way, and at best make continuous delivery more effective.

As you developed your application architecture, were there places where you saw concerns from different domains? How did those affect your application architecture? Did your application architecture evolve as you went through the exercise?

Be prepared to share your experience and observations with the class.

Lab 7 - Technical Architecture

Technical Architecture describes the platform that an application is built for and deployed on. The solution architect, either on their own or cooperating with specialists, describes how the application will actually be made to operate.

A good technical architecture should meet quality of service requirements, but do so in ways that are accepted in the organization. In most enterprises, there are well-understood patterns for technical components. In general, a solution architecture should expect to be making use of existing patterns, rather than developing technical solutions from scratch. Even if there are no established standards, a good solution architect should seek out and follow industry best practices and known solutions.

Part 1 - Outline a Technical Architecture

Outline a technical architecture for your solution.

Use whatever level of detail you find appropriate, but be prepared to show how your technical architecture supports

- The packaging of application components you identified in Lab 6
- The Quality of Service requirements you identified in Lab 4.

Part 2 - Outline a cloud-based Technical Architecture

You may have already done a cloud-based technical architecture in the last part of this lab. If you did not and if you have have the time, sketch a cloud-based technical architecture.

Part 3 - Retrospective

How did your technical architecture and your application architecture align? Were there things in your technical architecture that make you rethink aspects of the application architecture?

If you did both a non-cloud and a cloud technical architecture, what did you feel are the major differences between them? Are there pros and cons for each?

Share your observations and findings with the class.

Lab 8 - Retrospective

This workshop looks back on what we have done in previous workshops. This will be an open-ended discussion – take the discussion where it leads you!

Think back on the workshops that you have done in the course. The workshops attempt to follow more or less a process that you can follow when developing a solution architecture. Some questions for discussion:

- Did activities flow naturally from one to the next? Did you feel that activities built on what you had done before?
- Imagine you are developing your solution architecture in the real world – based on your observations of the workshops, what activities would you expect to iterate over?
- Where did you find gaps or "bumps in the road" – places where activities did not flow naturally from one to the next? What do you think those challenges would look like in the "real world"?
- Think about solution architecture work that you have done in the past – do you have examples you can share where you ran into bumps in the road? What happened? What was the impact? What did you do to overcome the challenges?

Now, think about the process of solution architecture in your organization,

- Thinking about activities that were explored in workshops, are there things that your workplace
  - Should start doing
  - Should think about *not* doing
  - Should continue doing

  Why?
- Do you have a well-defined process with clearly defined deliverables? What would you change and why?

Lastly, if you could tell your colleagues one or two things that you think would benefit solution architecture at your company, what would you say?

Again, don't restrict yourself to these questions!

Lab 9 - Example Problem – Acme Paint and PaintPro

Part 1 - Acme Paint

Acme Paint is an established manufacturer and supplier of paint to the construction industry. They serve midsize to large customers and operate largely as a business-to-business enterprise, with established customer relationships, dedicated account managers, and tightly focused marketing.

Acme Paint has been successful and is a major paint supplier to the construction industry. That success, however, limits Acme's growth; with a dominant market share, acquiring new customers is increasingly difficult and it is challenging to maintain Acme's margins.

Acme's leadership team has decided to grow through expansion into an adjacent market segment - consumer paint.

Acme does not wish to invest the capital required to establish a retail presence. Instead, Acme intends to sell through wholesalers - primarily hardware stores and renovations centres that may carry multiple brands of paint. To differentiate itself from competitors, Acme will position itself as offering a superior product ("Construction-tough!") in conjunction with added-value services that help customers to make optimal paint choices for their needs. These services include matching the type and grade of paint to the specific use case, and (most importantly) matching colour specifications.

Acme's go-to-market strategy has a number of components, including onboarding retail partners, extending its supply chain and developing brand awareness through consumer marketing. Acme's strategy also depends on developing and delivering applications to help consumers with product selection and colour matching.

Part 2 - Mandate

You have been tasked with leading the development of a solution architecture for PaintPro, an application that Acme Paint would like to offer to retailers carrying Acme Paint products.

PaintPro will be used by retailers to help sell Acme Paint products; by offering sales aids, customer-oriented information, and colour-matching services, PaintPro encourages retailers and customers to choose Acme Paint products over competitors.

PaintPro will serve two main functions. First, PaintPro will make relevant information available to retailers and customers. The application should help find and deliver relevant, actionable information, providing a service that competing brands lack and making sure retailers and customers are aware of Acme Paint's superior product.

PaintPro's other function is to help move customers from competing brands to Acme Paint products. Major paint manufacturers create names for their products and paint colours that are enticing to customers, but help lock the customer into a brand because

colour names are proprietary and cannot be readily compared to other brands. PaintPro will offer aids to identifying a matching Acme Paint colour.

Customers who are moving from competing brands will often need to find an Acme Paint colour that matches existing paint, but they may not know the manufacturer or colour name of the existing paint. To match a colour precisely, they would need to measure the colour of the existing paint; to do so accurately requires a hardware device called a "colorimeter". The PaintPro application is expected to have functionality to maintain a small inventory of colorimeters at participating retailers, and keep track of those devices when they are loaned out to customers. Further background information about paint colour naming and paint colour measurement is provided below.

In addition to supporting customers, PaintPro is also to support retailers who carry PaintPro products. Sales aids, sales training, product information, and sales associate training material should all be made available through the PaintPro product. Sales associated should have some information presented to them proactively, and also have the ability to search for and retrieve relevant information.

PaintPro should also be able to inform retailers about product availability from Acme Paint; PaintPro is not intended to replace a retailer's order management system, and is not concerned with orders that the retailer places with Acme Paint, but PaintPro does support buyers with product availability and pricing, particularly if there are manufacturer promotions.

The audience for the solution architecture includes

- Management - Who want to understand the functionality of the application and assure themselves that it will meet their business objectives.
- Architectural governance - Who want to verify that the solution architecture meets Acme Paint guidelines and requirements.

  NOTE For the purposes of these labs, we do not specify any particular requirements for architectural governance; don't worry about delivering anything specific for architectural governance, just keep in mind this is often a requirement. In the real world, architecture governance processes should specify what needs to be provided in the solution architecture.
- Development - Who will use the solution architecture as the basis for software engineering and development.

The solution architecture will consist of a number of deliverables, which will be outlined in each of the associated labs.

To keep the lab work manageable, each lab is treated as an independent artifact; you do not need to worry about producing a solution architecture document or any such consolidated document.

Part 3 - Business purpose of PaintPro

Allow retail staff to look up information about Acme Paint products (staff pull information)

Newsletters

Paint specifications

Safety and composition information

Sales aids

Provide retail staff with updates and bulletins from Acme Paint (push notification)

For information, updates, and bulletins, provide a means for Acme Paint staff to create and manage content.

Provide a product recommender to match Acme products to customer needs

Interactive questionnaire

Or a form that customers complete

Provide a mechanism to determine the right Acme paint colour for a customer

By matching an Acme Paint colour to the colour name of a competitor's paint

By matching an Acme Paint colour to a photograph of the desired colour see backgrounder on colour-managed photography for what is required for this to work

Exclusions

PaintPro does not manage inventory or orders Those functions are presumed to be satisfied by retailer's enterprise systems

Part 4 - Acme Paint business environment

Acme Paints is headquartered in Metropolis, in the United States. It has several paint manufacturing plants across the United States and Canada, with established distribution networks through third-party transport that reach major markets in both countries. There are three regional offices servicing Canada, Eastern US, and Western US.

Product development is centralized in the headquarters in Metropolis. Marketing, which is responsible for creating customer-facing content as well as running marketing campaigns, is distributed across head office and regional offices.

Sales representatives, who deal with business clients and are expected to support retail resellers in the future, are distributed across the country and work out of home offices.

Part 5 - Acme Paint technical environment

Acme Paint owns and operates two geographically separated data centres. Enterprise applications are typically implemented in a hot-warm deployment model, with a primary application instance and a failover instance in the other data centre.

Enterprise data is managed using MariaDB. MariaDB is a fork of MySQL and is an open-source enterprise-class relational database. (For the purposes of the lab, we want to specify a classic relational database. We picked MariaDB because it deserves more attention in the enterprise space.)

Acme is making increasing use of cloud to augment its data centres. Enterprise applications and most master data are still located on-prem, but cloud is being used increasingly for new applications. (For the purpose of the lab, it does not matter which cloud provider Acme uses - pick whichever you are most familiar with if you want to use any vendor-specific terminology.)

Applications supporting business customers are transitioning from on-prem deployment to cloud, and all new customer-facing applications are built for cloud deployment.

On-prem applications are primarily built in Java. Web applications are primarily built in Node, but Acme has not settled on technical platforms for new applications and permits development in any platform that can be justified as fit for purpose. (For the purposes of the lab, we don't want to restrict technology choices if participants have a preference.)

Lab 10 - Background information for Acme Paint (optional)

Part 1 - Colour spaces and colour naming

A colour can be specified by combinations of numbers. There are several different schemes fro numerically describing colour, called “colour spaces”. A colour space describes what colours can be numerically represented and what the numbers mean. Common colour spaces include

- RGB Representing colour as three numbers, one for each of the intensity of red, green, and blue. Suitable for describing the colour of something that emits light, such as the pixels of a computer monitor.
- CMYK Four numbers, representing the density of cyan, magenta, yellow, and black. Used to describe the colour of objects that reflect light, such as print on a page. Cyan, magenta, yellow, and black are the most common colours of ink used in colour printing
- CIELAB A colour spaced based on how the human eye sees colour and widely used in professional applications. CIELAB is often called “lab colour”, though the “lab” does not refer to “laboratory”; the L, a, and b in the name are three parameters, L being the luminosity (brightness), a and b being coordinates in a colour space.

A properly defined colour space is not just a set of colour numbers - it also includes precise definitions of the meaning of the numbers, their minimum and maximum values, and how certain standard reference colours are defined. RGB, for example, isn’t really just red, green, and blue. The author’s Mac, for example, uses a colour space called “Adobe RBG (1998)” for its external monitor.

In the paint industry, colour specification and colour matching are critically important. If I purchase a litre of “Sky Blue” paint from Acme Paints, I want to be able to buy another litre next year and know it will be exactly the same colour. A paint manufacture goes to great lengths to ensure this through control of the manufacturing process and pigment mixing specifications that retailers use when mixing a can of “Sky Blue” paint for a customer.

Paint manufacturers don’t use colour spaces, at least not with consumers. A consumer wants “Sky Blue”, not “R=140, G=205, B=246”. The manufacturer will give names to the colours that it offers to customers. For the purpose of this lab, we will call the collection of colour names a “colour name-set”.

Each manufacturer will have a unique colour name-set. Acme Paints and Bravo Paints may have paints that are visually the same colour, but Acme Paint could call theirs “Sky Blue” while the effectively identical colour from Bravo Paint could be called “Summer Sky”.

Part 2 - Measuring colour – Digital images

Since almost everyone has a smartphone with a high-quality camera, could we measure a colour simply by taking a picture of it? We could, but the measurement would be surprisingly inaccurate. The colour the camera sees is strongly affected by lighting; imagine how different a room looks in morning sun, in twilight, and when lit by lamps at night.

Smartphone cameras make colour measurement even harder by trying to compensate for lighting. Smartphone pictures are usually heavily processed by algorithms in the phone to make them more attractive, but this makes the unsuitable for accurate colour measurement.

A smartphone photograph can be used for reasonably accurate colour measurement if the photograph includes standard known colours that can be used as a reference. A colour measurement algorithm can see how the known colours appear in the photo and use that information to compensate for lighting and image processing. Standard colours are available in a "colour card", which is a physical card printed with known standard colours, manufactured under controlled conditions for consistency. Colour cards have long been used, and are still used, in professional photography and are widely available.

Part 3 - Measuring colour – Colorimeter

A colorimeter is a device used to precisely measure colour. Professional instruments are used in manufacturing, quality controls and design in a wide range of industries, from automotive to cosmetics. Recently, consumer-grade colorimeters have become available with costs as low as 100$USD retail. These are often pocket-sized, battery powered, and connect to a smartphone by Bluetooth. They make precise colour measurement available for commonplace tasks like monitor calibration, small-batch manufacturing, and art production.

A colorimeter produces a colour measurement, using one of the colour spaces described above. The colorimeter is more accurate than a digital camera as it has precise control over the light source (which is part of the colorimeter) and the sensor.

Consumer-grade colorimeters typically have libraries that allow applications to integrate directly with the colorimeter; some products, such as the Nix Mini 2 Color Sensor, are developing an ecosystem of applications that make use of the inexpensive hardware. (www.nixsensor.com)