Design Your Software Architecture Using Industry-Standard Patterns

4 hours Hard

Last updated on 6/29/20

Data-Centered Architecture

In this chapter, we are going to cover the data-centric architecture, what is it, and why it is useful in certain situations. At the end of the chapter, I will show you a real-word example (STU National University), and then you will solve a similar case applying what you've learned. Are you ready?

What Is a Data-Centric Architecture?

Have you ever bought an airline ticket online? Sometimes when you complete your purchase, the website recommends a hotel, car rental company, and even more options. Sometimes these offers continue to pop up over the following days (by email), or after you have finished your trip.

The airline has a selling strategy: it identifies different types of customers and defines different actions to do with each customer after the purchase operation. Here is an example of what a strategy like this could look like:

Customer Type	Definition	Next steps after the purchase
1	Between 21 and 35 y.o. Travels solo. Paid with premium credit card.	• Offer 5-star hotel. • Offer top of the line rental car. • Offer Caribbean Resort stay with discount. • Offer restaurant discount coupon.

2	Between 21 and 35 y.o. Travels with companion. Bought two tickets on the same flight.	• Offer 5-star and 4-star hotels for future trip. • Offer 4x4 rental vehicle. • Offer restaurant discount coupon.

This pattern continues for customer types 3, 4, and 5 to 20.

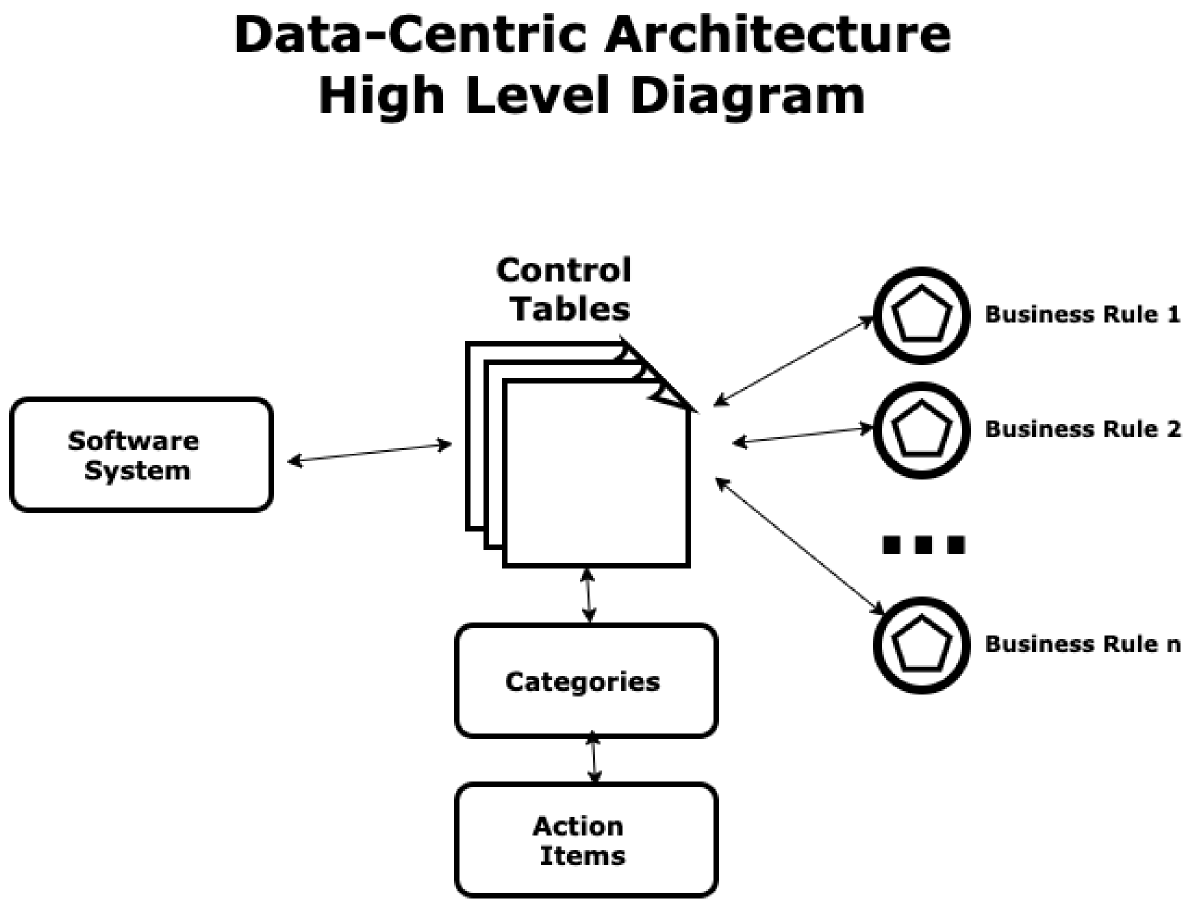
Now, let's suppose that the airline marketing strategy changes, and you no longer have 20 customer types, but 25, and some updated “next steps” for after the purchase.

Are you going to rewrite the code of the sales module in your software system every time the marketing strategy changes? Not at all. A good practice is to include a table like above in the system database and make the software reads the table and acts accordingly for each customer type. In this way, if the marketing strategy changes, you will only have to change the table, not the code.

This is how a data-centric architecture works. It uses **table-driven rules** (as opposed to code-driven rules) to manage different categories and actions to come after a certain event. These tables are called control tables, and they allow programs to be simpler and more flexible.

What Is the Structure of the Data-Centric Architecture?

Let’s see what this looks like:



Data-centric architecture

As you can see in the diagram above, a standard data-centric architecture has five parts:

- **Software system:** The system developed using the data-centric architecture model.
- **Control tables:** A set of tables that define the action items the system must take for each category.
- **Categories:** A set of categories of an item in the system (customer types, product types, item types, employee types) used to execute action items.
- **Action items:** A set of actions to be executed for a certain category.
- **Business rules:** Business rules that determine the behavior of action items upon categories.

If you have a specific group of people that can define the categories or action items or both, you should add them to the diagram. Usually, this is represented by creating a box with their name and linking it to categories or action items via an arrow.

How does this work in practice?

- We have a software system that sells airline tickets online.
- The system has a set of control tables that implement the sales strategy of the company.
- There are many customer categories and action items for each category (you can see a detail of these categories in the prior customer types table).
- These definitions respond to business rules that come from the sales strategy.

There are several **advantages** to using data-centric architecture:

- If categories change or are added, we do not need to change code.
- Business rules are much easier to maintain in a table than in a group of programs.

There are also a few key **disadvantages:**

- This scheme may have performance problems because each time the code runs, it must look up data in a table.
- If the control table gets damaged, the whole system doesn't work.

When Would I Use Data-Centric Architecture?

So, what are some situations that this would be useful for?

Example Name	Definition	Real-World Example	Advantages
Airline CRM System (Customer relationship management)	CRM is a software system that manages personalized relationships with a company's clients. Marketing campaigns, customer service, and cross-selling campaigns are usually managed with a CRM system.	• Oracle E-Business Suite (oracle.com). • Microsoft Dynamics (microsoft.com.	A CRM system can take care of millions of customers. It would be impossible to do this manually. CRM systems were introduced in the '90s, and have increased sales productivity significantly.
Commission payment module	Usually part of a sales system, a commission payment module is an application that calculates commissions for salespeople and arranges the corresponding payments in a predefined period.	Any sales system has a commission payment module.	Commission payment modules usually have a commission type table that determines the commission calculation and represents the

company policy for paying sales commissions.

Case-Study: Solving a Business Problem With Data-Centric Architecture

Let's apply what you've learned about data-centric architecture and see how it works in a real organization.

STU is a national university. Here are some **quick facts:**

- STU has more than 45,000 graduate and undergraduate students.
- STU is updating its software system for new students, diplomas, tuition administration, and loan administration.

What's the Business Problem?

STU wants to define action items for student categories based on certain rules. These action items come from a communication policy established by the board of directors. For example, "If a student loan is approved, then we must send an email to the student, their parents or tutors, and update our administrative tuition records accordingly."

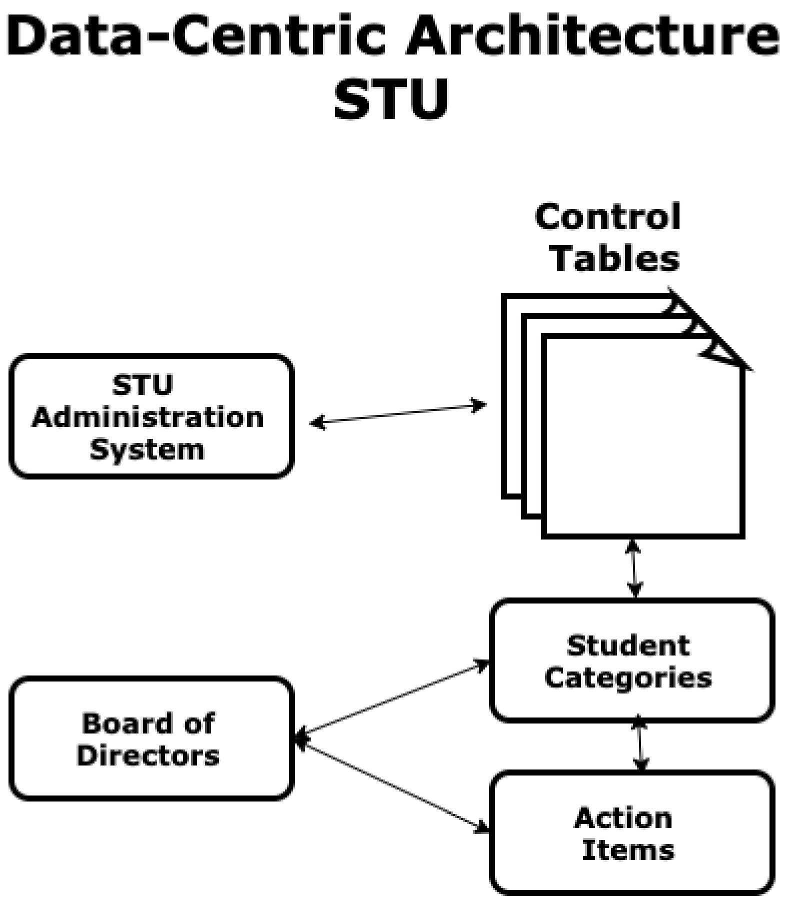
Let's analyze this situation!

1. Imagine that there are many student categories at STU: graduate, undergraduate, a student that applied for a loan, a student whose loan has been approved, a student aged 16 to 21, a student whose age is over 21, an international student, etc.
2. A student can belong to multiple categories.
3. Action items defined by the board are executed for a group of student categories, usually, if they happen together. For example, "If the student is foreign **and** applied for a loan, then do X, Y, Z."
4. We do not want to rewrite the code each time the board updates or creates action items for a certain category.

It's a perfect scenario for a data-centric architecture: there must be a table for action items upon categories, and the system must look up each table entry in order to act.

What's the Solution?

Now, let's see the detailed architecture diagram for this solution:



STU data-centric architecture

Let’s explain each component of the ERP system:

- **Administrative system:** This is the system developed for STU.
- **Control tables:** This is a set of tables that define the action items the system must take for each category according to board decisions.
- **Board of directors:** This is a group of directors, trustees, or advisors in a university or college that defines the strategy of the institution to enroll students and maintain an adequate curriculum and a desired level of quality.
- **Categories:** This a set of student categories.
- **Action items:** This is a set of actions to be executed for a certain category.

> **Try it out for yourself!**
>
> Now that you’ve seen one solution see if you can apply what you’ve learned to another!
>
> **Context:** Let’s imagine you’re a software architect at a major mobile operator. You have been asked to develop a software system to manage customer relationships with certain

customer categories such as residential, business, NGOs, government, new customer, new prospect, individual, organization, etc.

Your Mission: Now, it's your job to create an architecture for them!

Here are some **key questions** you can ask yourself to get started:

1. A customer can belong to many categories. How are you going to define the action items for a group of categories?
2. Who should you ask in your organization about customer categories? Who should update developers on marketing policies change?
3. What happens if action items change according to new marketing policies in the company?

Can you produce an architecture diagram for this business setting?

Once you've written out your version, check it against my **solution**.

Let's Recap!

Architecture Model	Description	Advantages	Disadvantages	When to use it
Data-centric	An application structure that uses data instead of code to make behavior decisions.	No need to analyze all cases in the code. Categories can grow indefinitely. Changes can be made without changing the code.	May get complicated to understand and maintain when there are many categories.	When you have many different categories and need to act differently for each one of them.

LAYERED ARCHITECTURE

REVIEW WHAT YOU'VE LEARNED

Teacher

José Esterkin

Software engineer, project manager and trainer. Director of Positive, a project management consulting firm based in Buenos Aires.