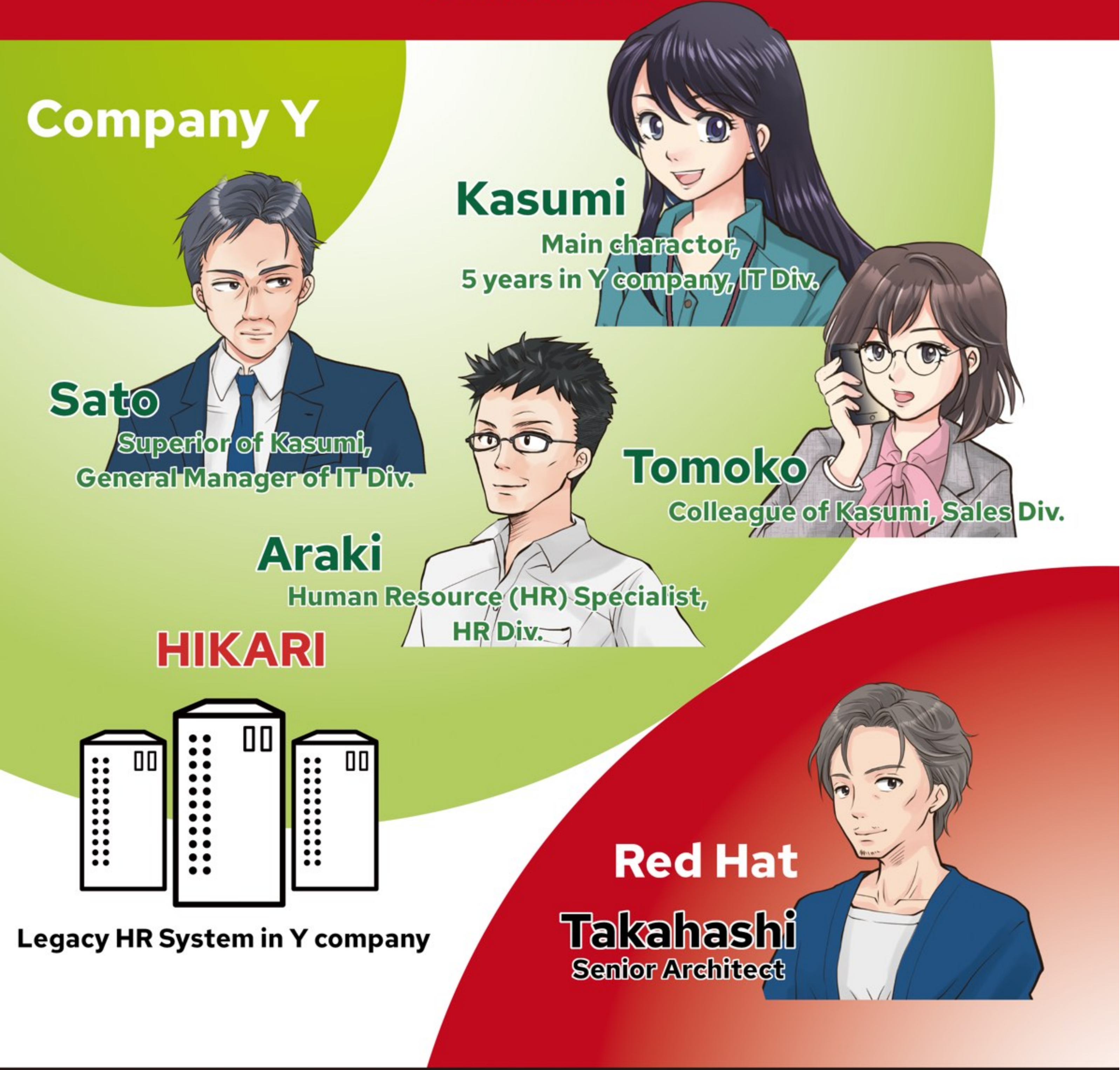
Rule Driven Development

A must-see for all developers!!
Methodology for Application Modernization

Red Hat

Charactor


Company Y
Kasumi
Main charactor,
5 years in Y company, IT Div.
Sato
Superior of Kasumi,
General Manager of IT Div.
Tomoko
Colleague of Kasumi, Sales Div.
Araki
Human Resource (HR) Specialist,
HR Div.
HIKARI
Legacy HR System in Y company
Red Hat
Takahashi
Senior Architect

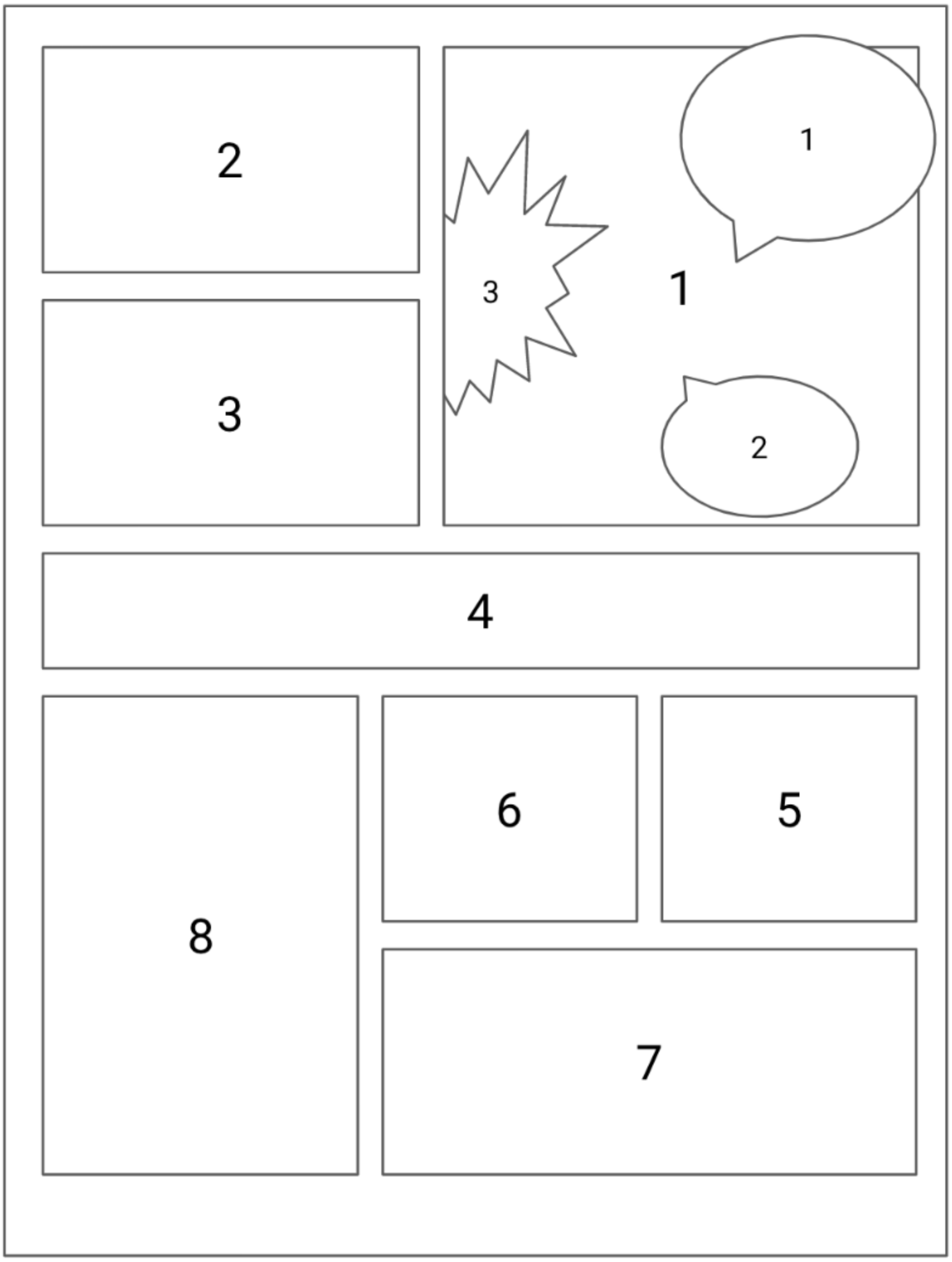
Story

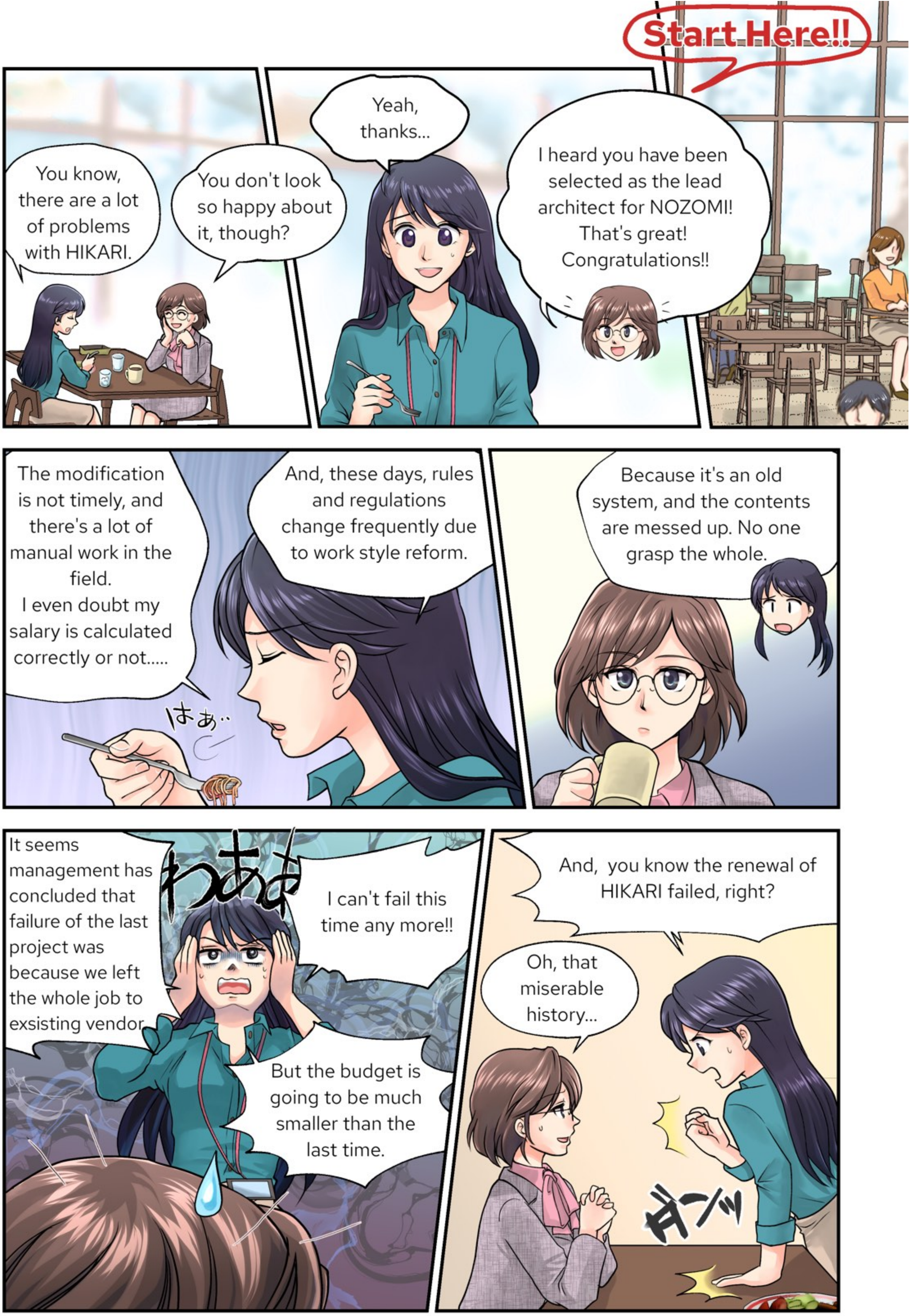
HIKARI is the system of HR in Company Y running from 20 years ago. Replaced into Open 10 years ago after running 10 years on Mainframe.
After that, the company continued to expand functions and replace DB. However because of consolidation of group companies, overall renewal plan has been established 5 years ago due to needs of system integration and increase of mantainance costs.
A team was formed with elite members of the IT div. and existing vendors, and began to analise the entire complicated HIKARI system. However , they could not finish analysis of current system after one year, and demands for new system accumulated significantly, and after all they could not even estimate entire renewal cost.
The renewal of current system was stopped, and no positive result at all, then finally the decision was made to cancel this plan in the 2nd year. In this stressful environment, many members got health issues or left company.
The cost already paid to the existing vendor was huge. After this failure, a contract was signed to extend support free of charge, and the legacy system will continue to perform in HR of Company Y.
However, the NOZOMI project has now been launched as re-renewal project of the HIKARI.

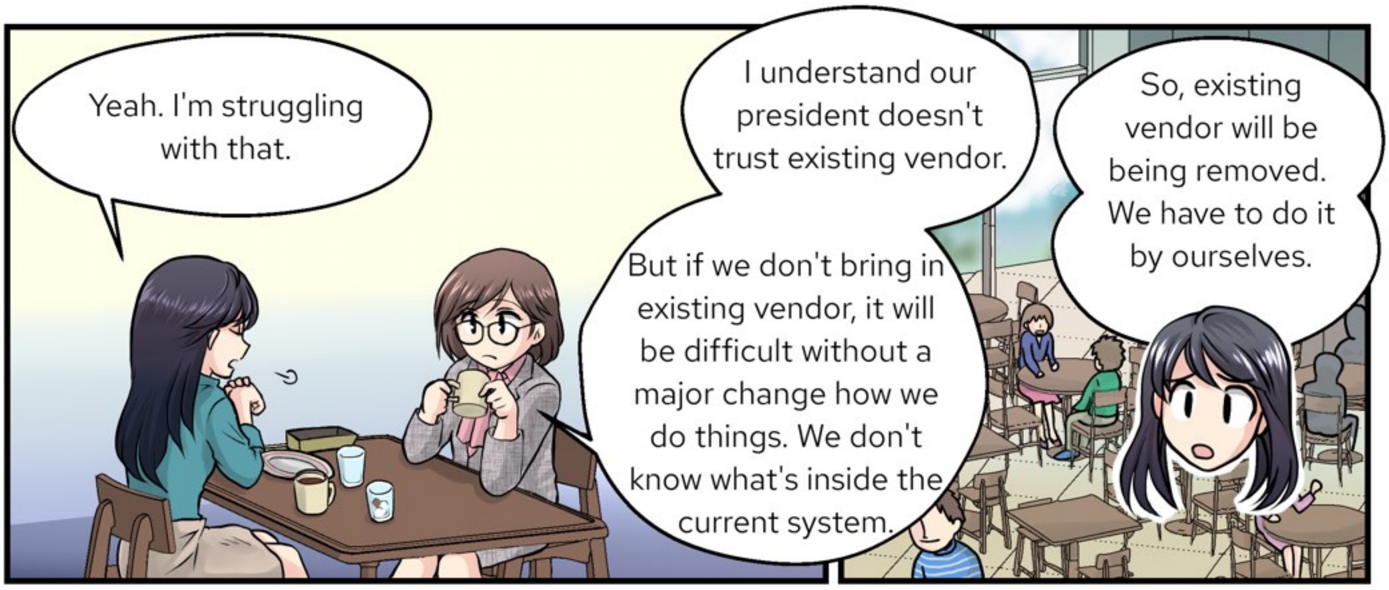
How to read manga

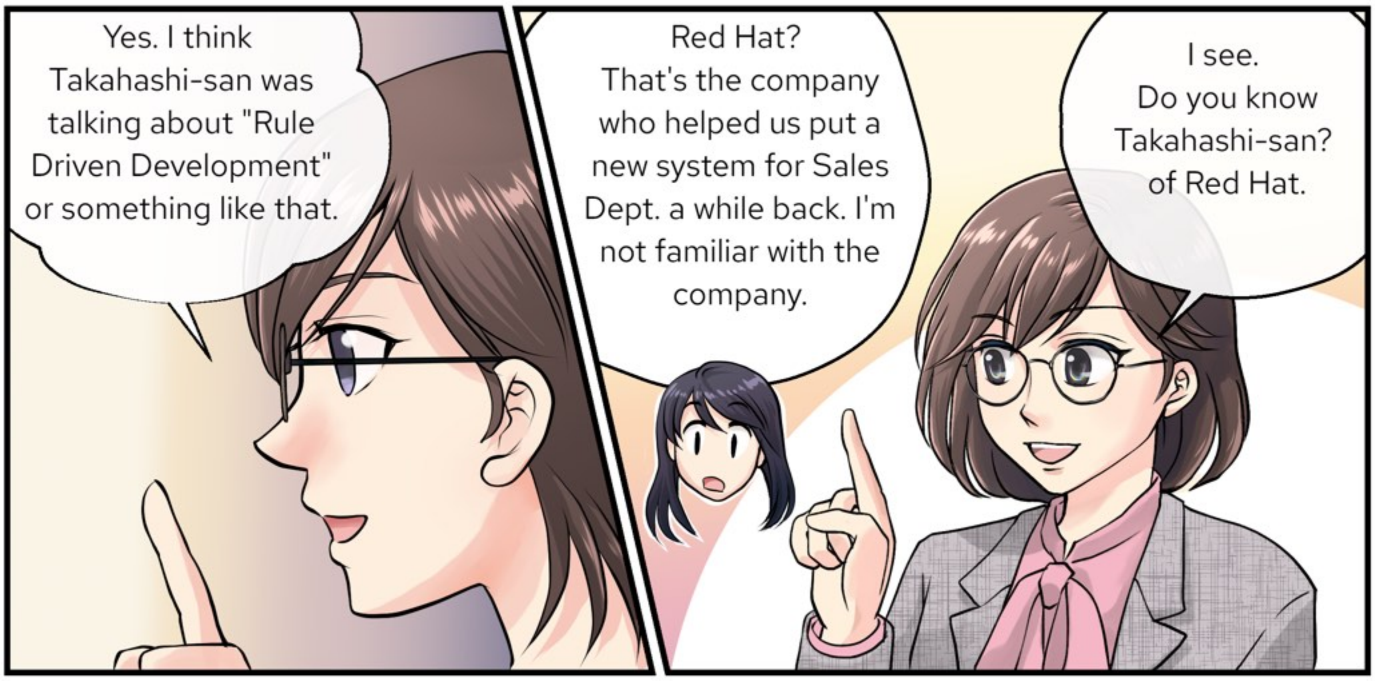
Friendly reminder, manga are supposed to be read from right to left, from top to bottom.

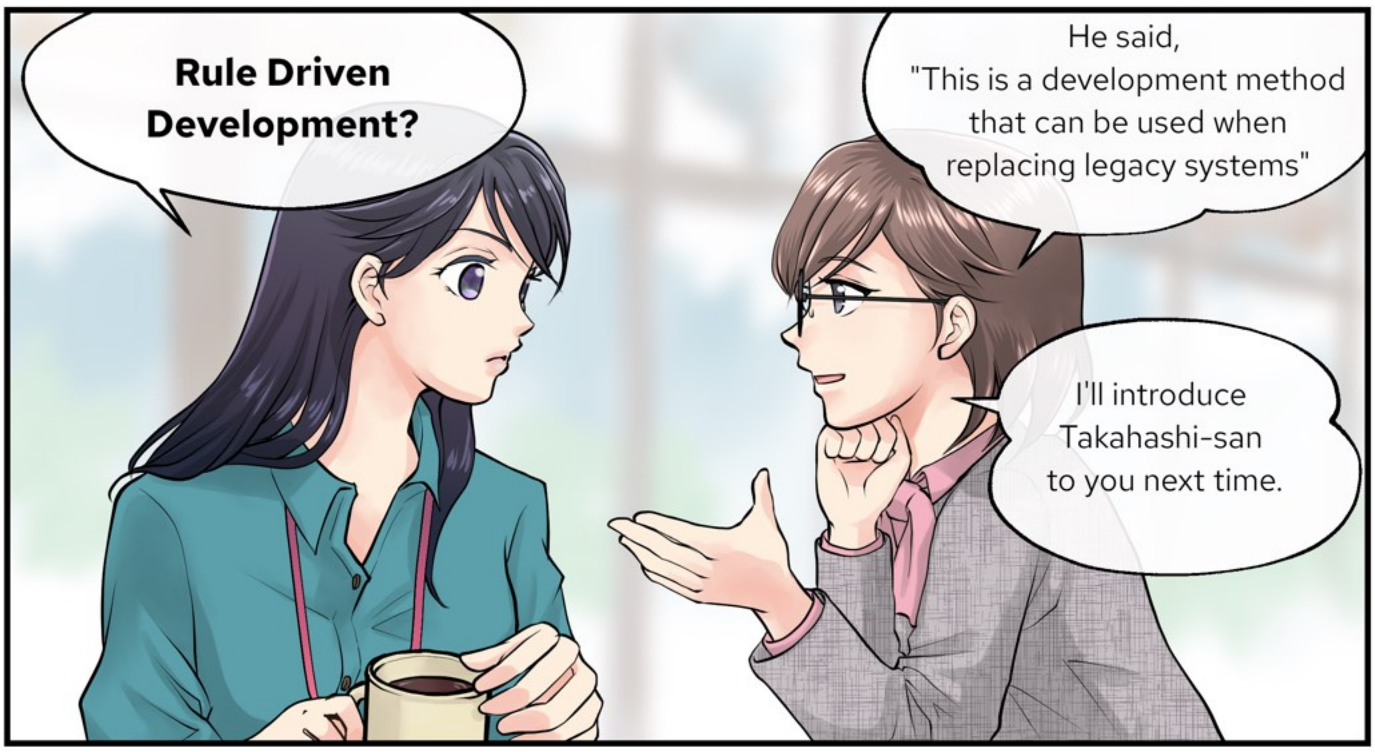
Example:

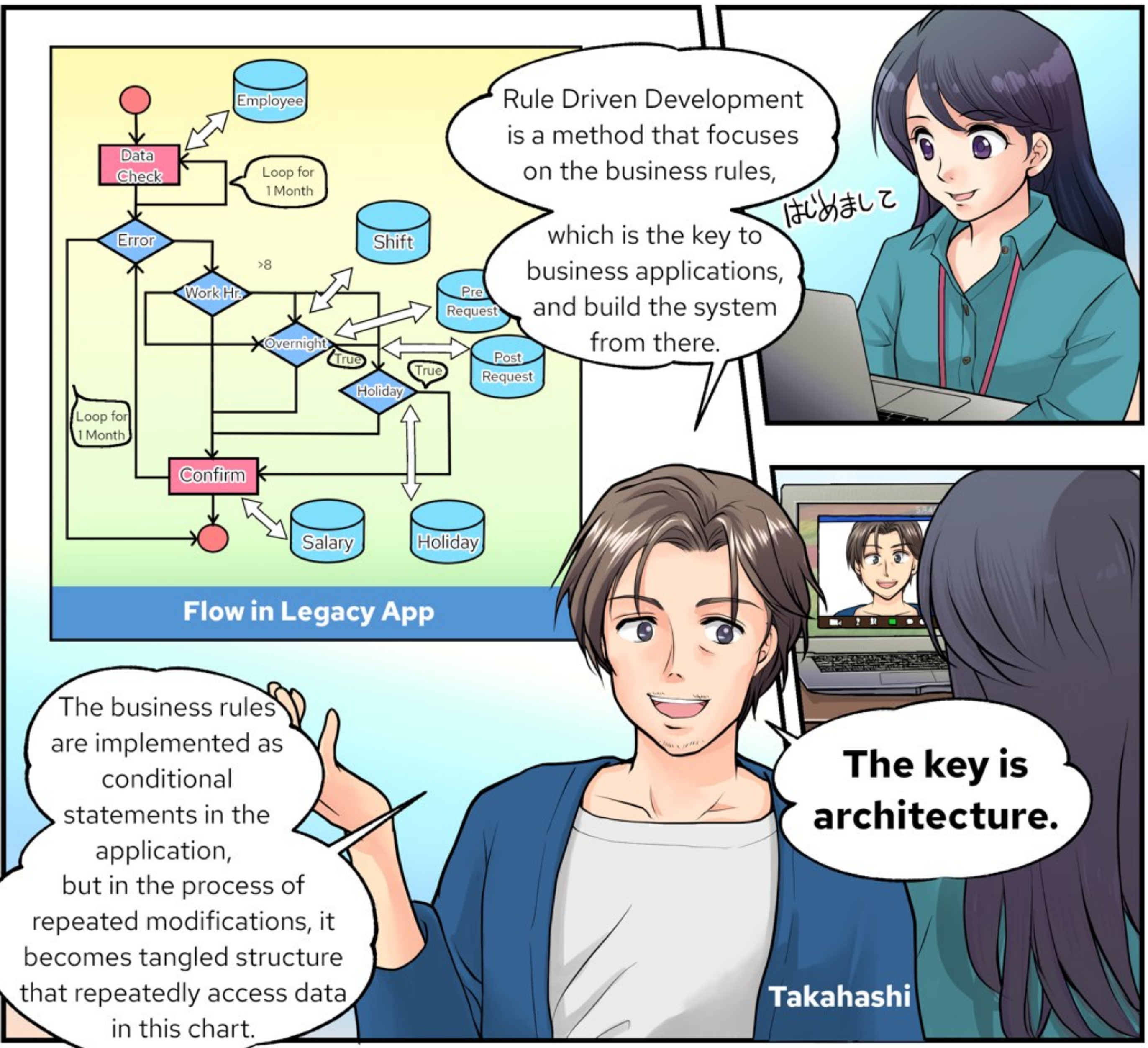


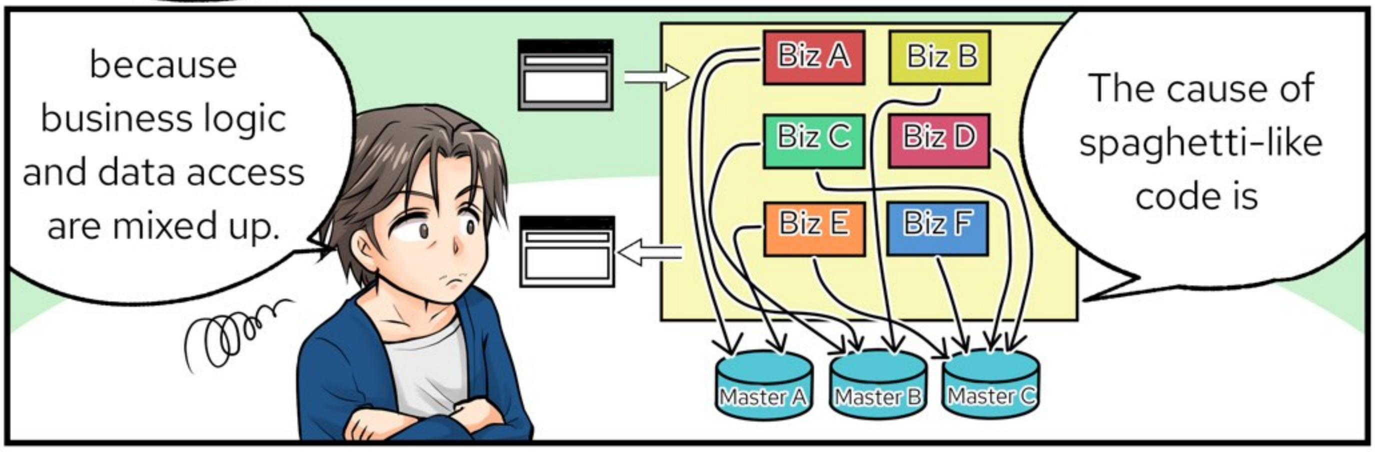

Start Here!!
I heard you have been selected as the lead architect for NOZOMI! That's great! Congratulations!!
Yeah, thanks...
You don't look so happy about it, though?
You know, there are a lot of problems with HIKARI.
The modification is not timely, and there's a lot of manual work in the field. I even doubt my salary is calculated correctly or not.....
And, these days, rules and regulations change frequently due to work style reform.
はぁ‥
Because it's an old system, and the contents are messed up. No one grasp the whole.
And, you know the renewal of HIKARI failed, right?
ダンッ
Oh, that miserable history...
It seems management has concluded that failure of the last project was because we left the whole job to exsisting vendor.
But the budget is going to be much smaller than the last time.
わあぁ
I can't fail this time any more!!

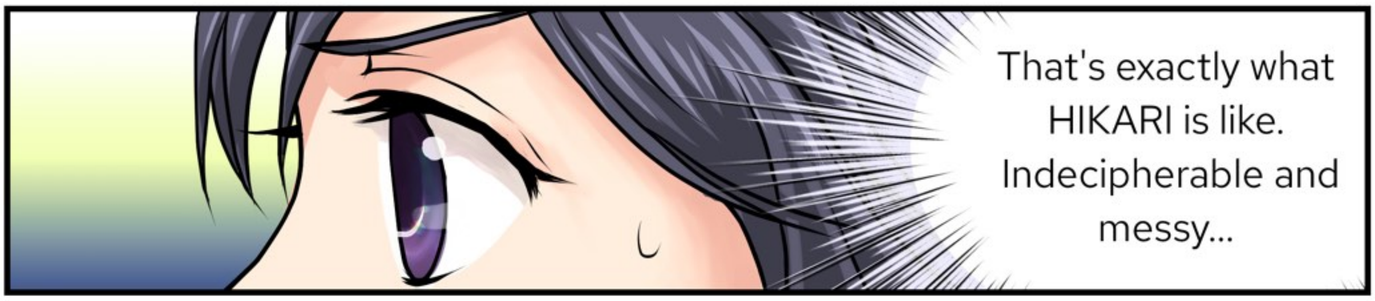

Yeah. I'm struggling with that.
I understand our president doesn't trust existing vendor.
But if we don't bring in existing vendor, it will be difficult without a major change how we do things. We don't know what's inside the current system.
So, existing vendor will be being removed. We have to do it by ourselves.

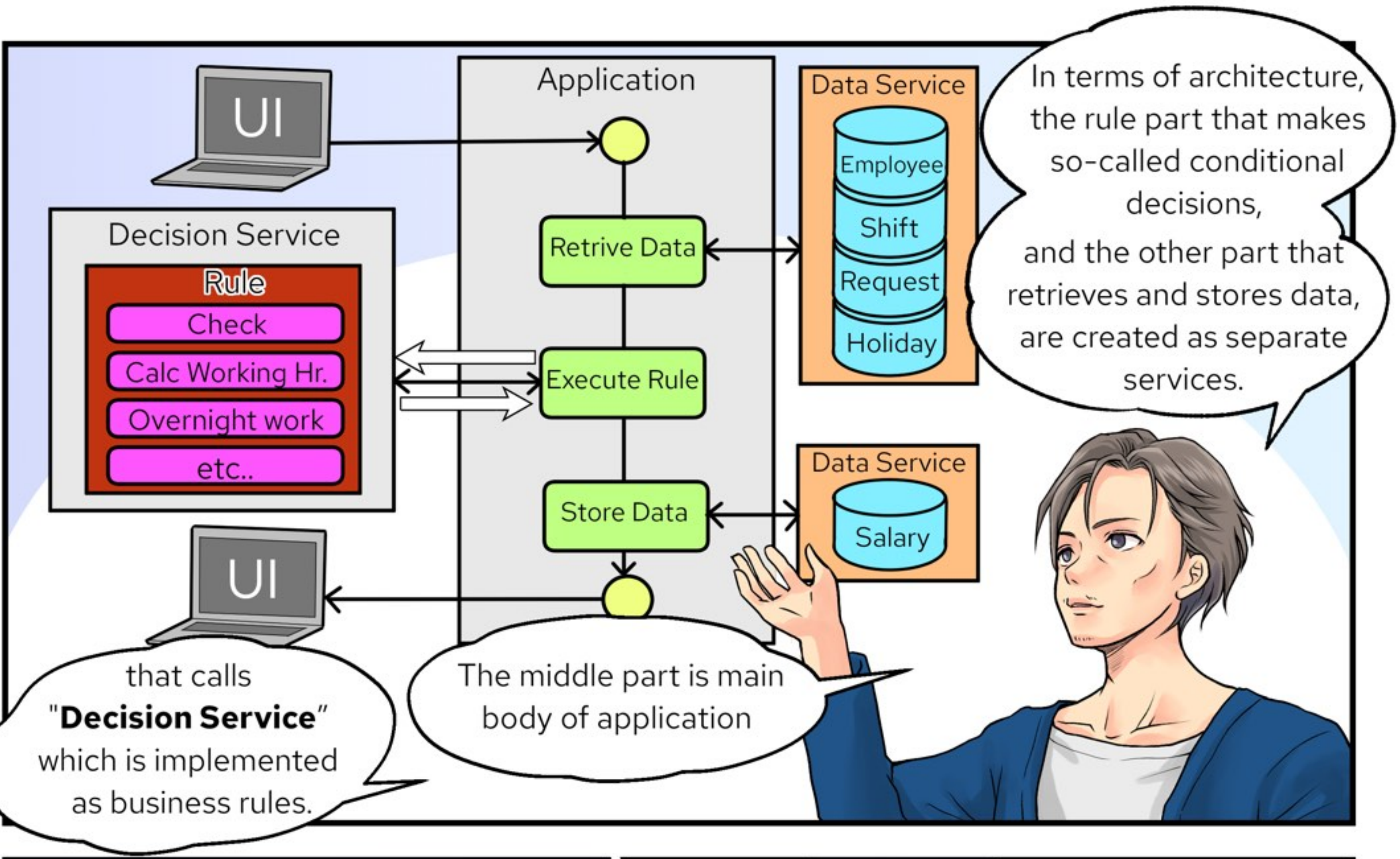

Yes. I think Takahashi-san was talking about "Rule Driven Development" or something like that.
Red Hat? That's the company who helped us put a new system for Sales Dept. a while back. I'm not familiar with the company.
I see. Do you know Takahashi-san? of Red Hat.

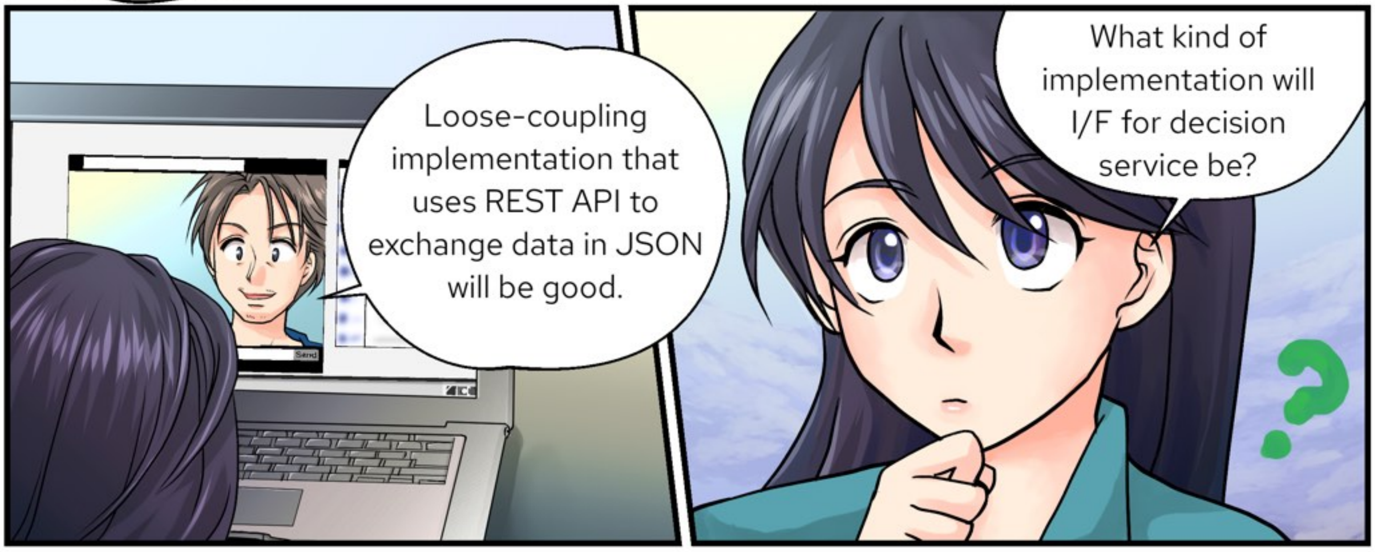

Rule Driven Development?
He said, "This is a development method that can be used when replacing legacy systems"
I'll introduce Takahashi-san to you next time.

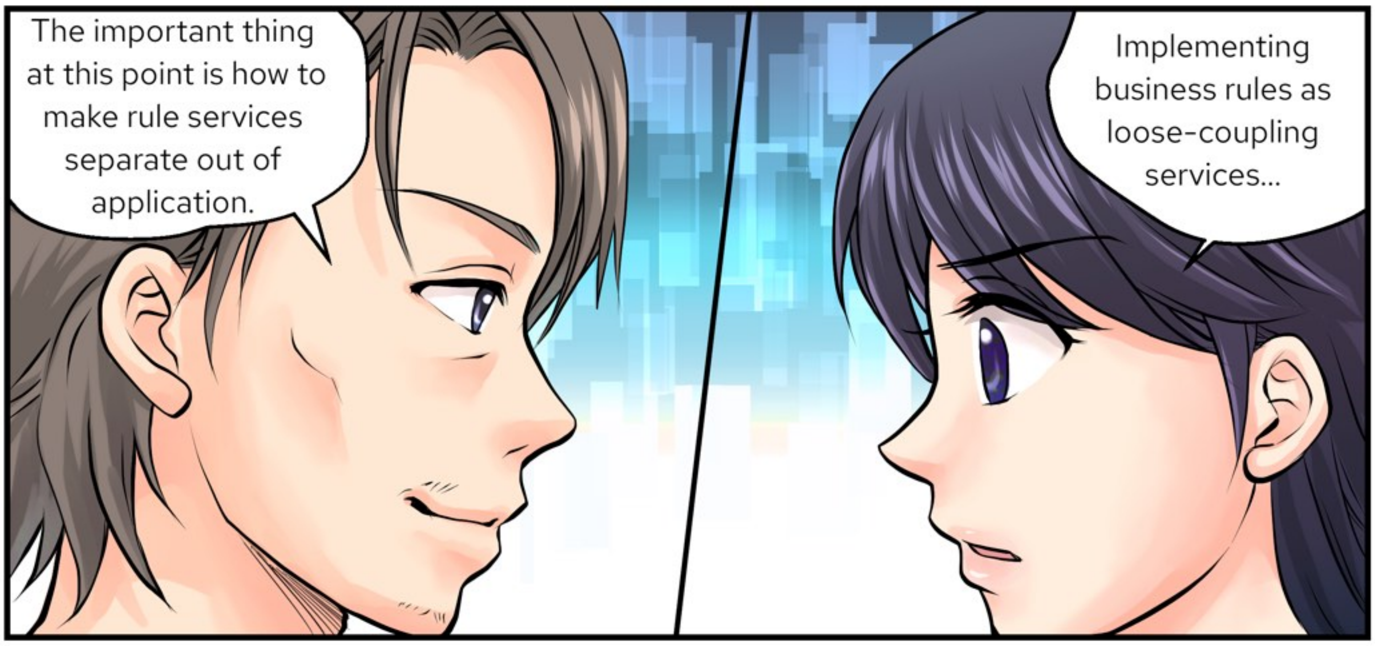

Flow in Legacy App
Employee
Data Check
Loop for 1 Month
Error
Shift
>8
Work Hr.
Pre Request
Overnight
True
Post Request
True
Holiday
Loop for 1 Month
Confirm
Salary
Holiday
Rule Driven Development is a method that focuses on the business rules,
which is the key to business applications, and build the system from there.
はじめまして
The business rules are implemented as conditional statements in the application, but in the process of repeated modifications, it becomes tangled structure that repeatedly access data in this chart.
The key is architecture.
Takahashi

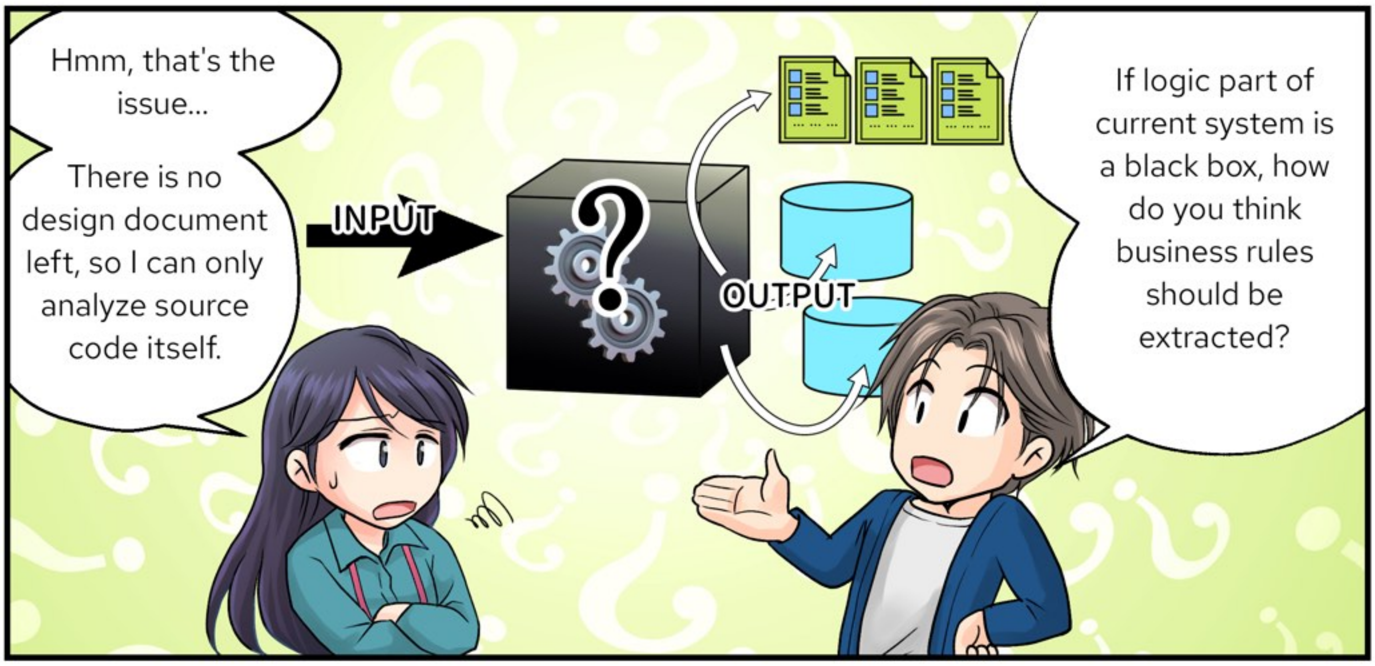

because business logic and data access are mixed up.
Biz A
Biz B
Biz C
Biz D
Biz E
Biz F
Master A
Master B
Master C
The cause of spaghetti-like code is

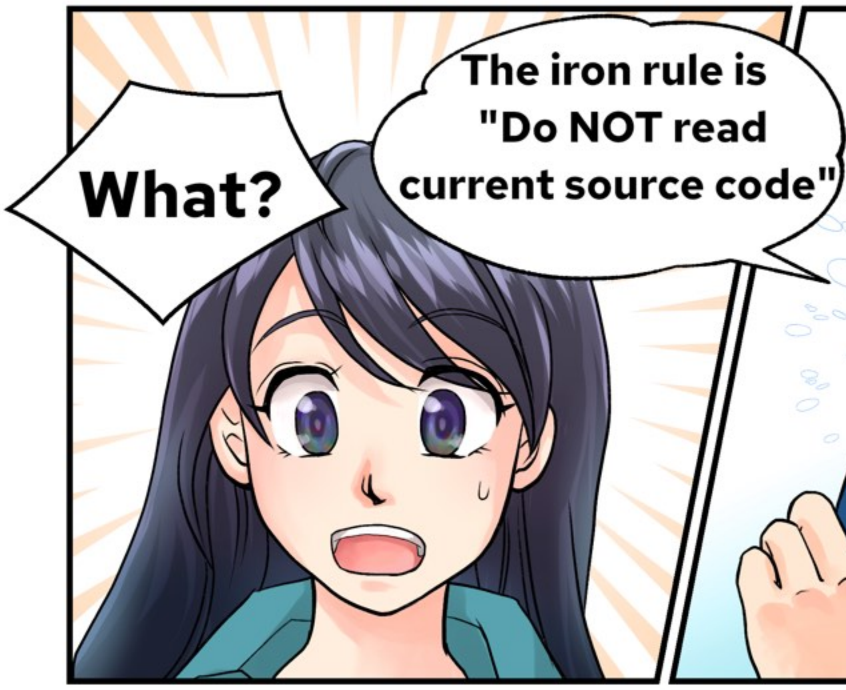

That's exactly what HIKARI is like. Indecipherable and messy...

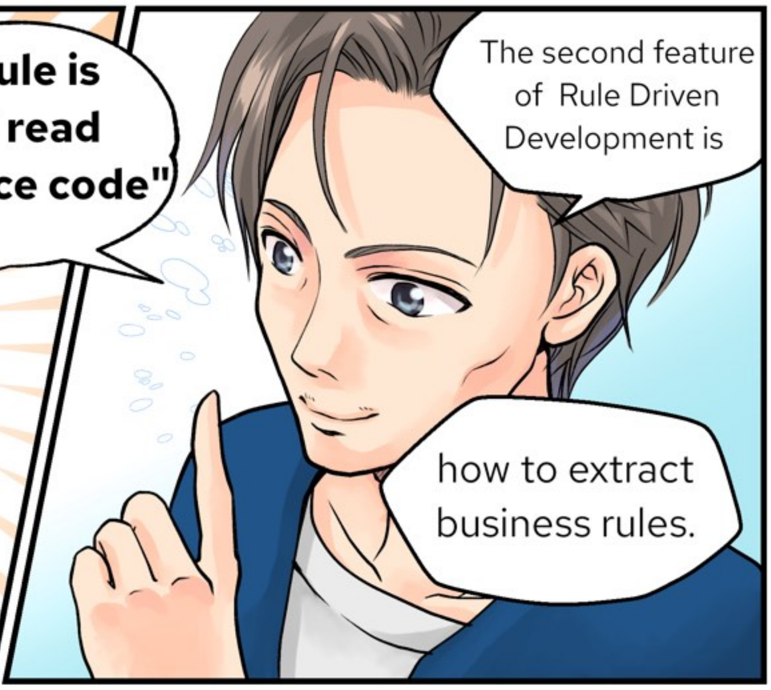

UI
Application
Data Service
Employee
Shift
Request
Holiday
Decision Service
Rule
Check
Calc Working Hr.
Overnight work
etc..
Retrive Data
Execute Rule
Store Data
Data Service
Salary
UI
In terms of architecture, the rule part that makes so-called conditional decisions, and the other part that retrieves and stores data, are created as separate services.
The middle part is main body of application
that calls "**Decision Service**" which is implemented as business rules.

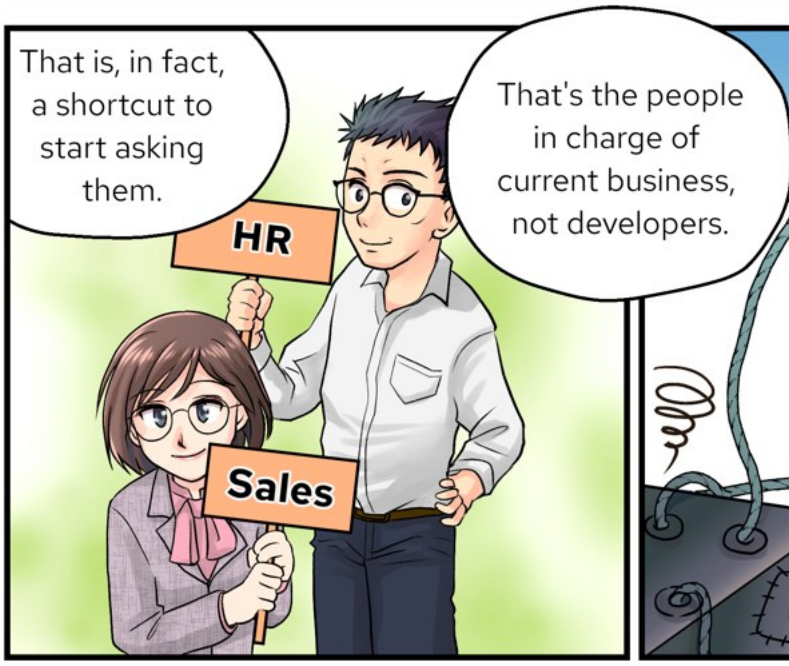

What kind of implementation will I/F for decision service be?
Loose-coupling implementation that uses REST API to exchange data in JSON will be good.

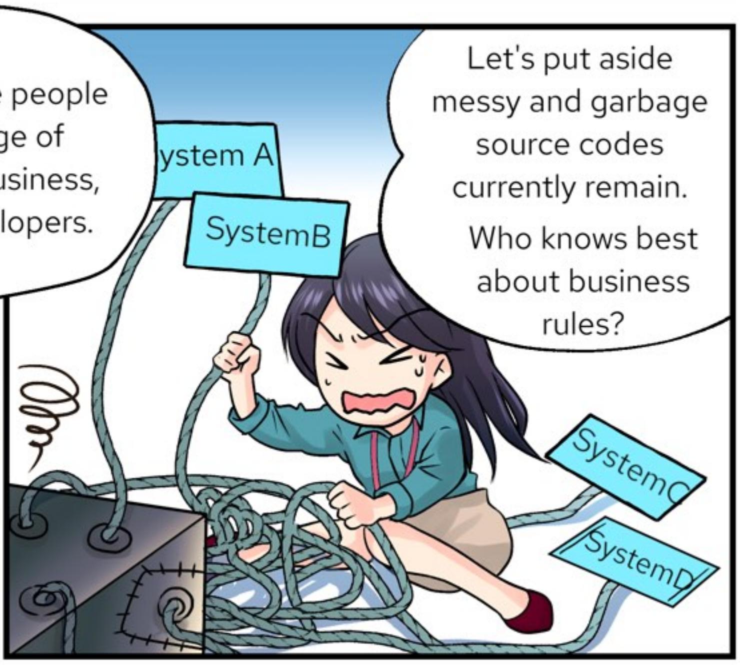

Implementing business rules as loose-coupling services...
The important thing at this point is how to make rule services separate out of application.

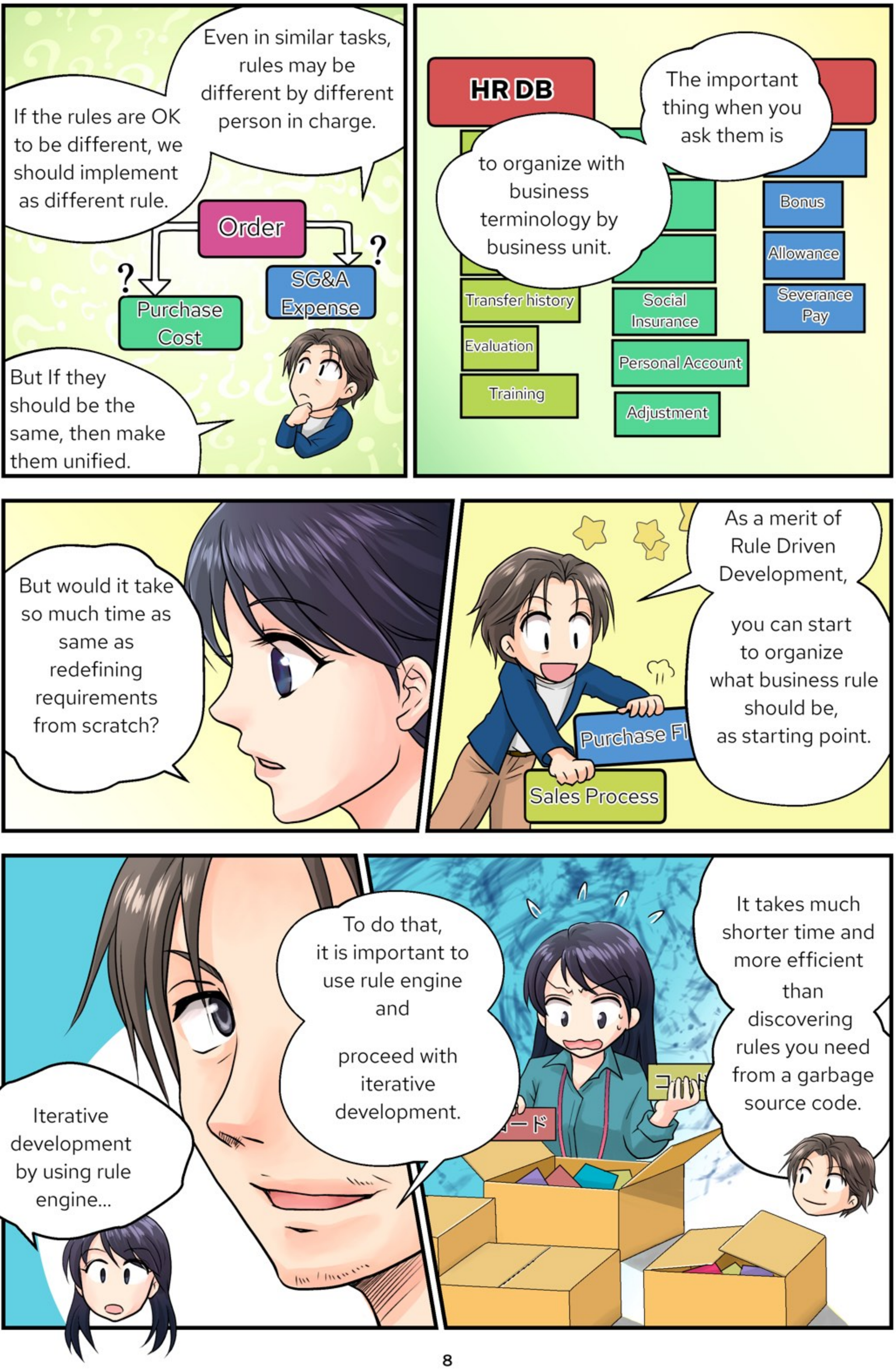

Hmm, that's the issue...
There is no design document left, so I can only analyze source code itself.
INPUT
OUTPUT
If logic part of current system is a black box, how do you think business rules should be extracted?

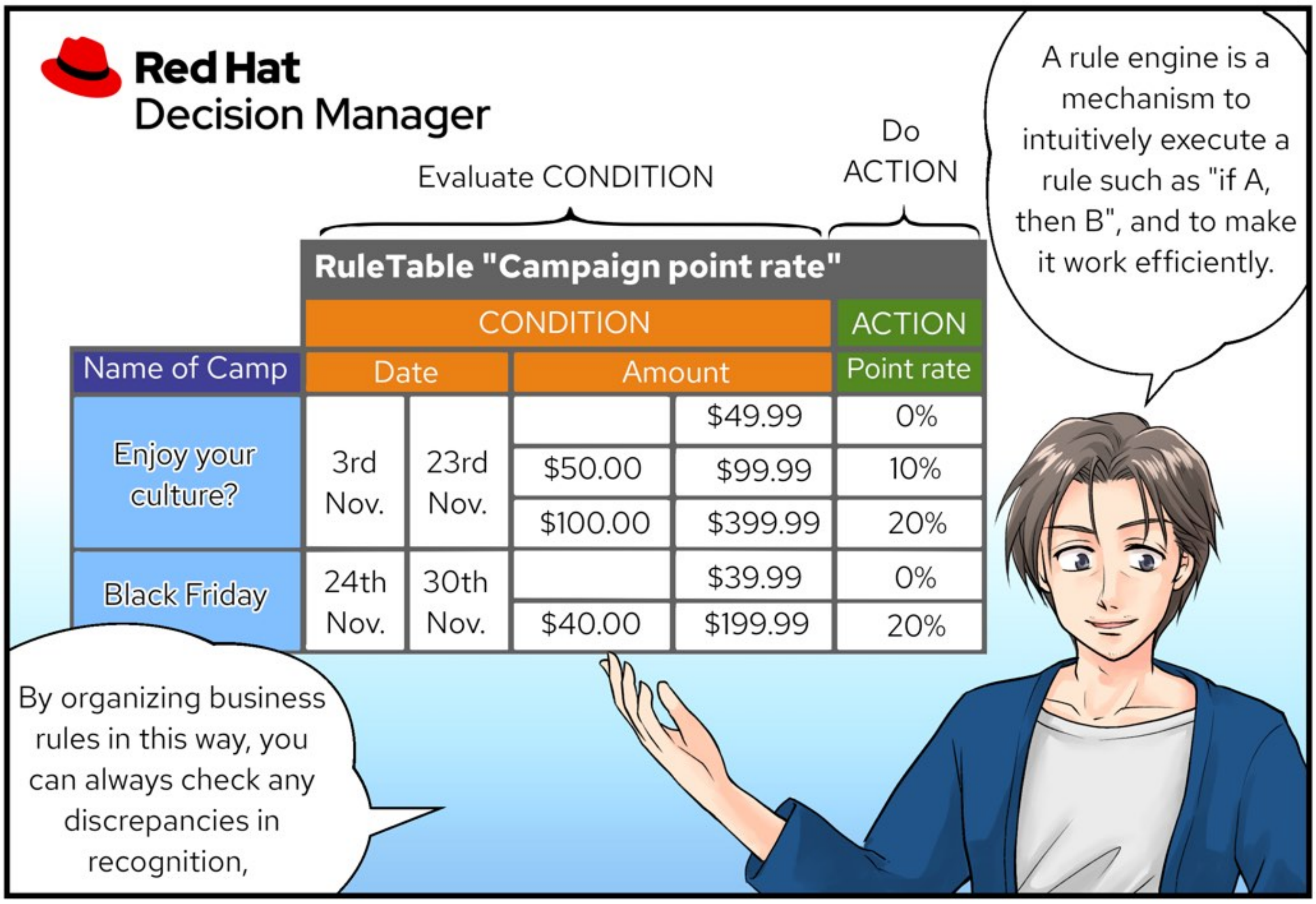

What?
The iron rule is "Do NOT read current source code"


The second feature of Rule Driven Development is
how to extract business rules.


That is, in fact, a shortcut to start asking them.
That's the people in charge of current business, not developers.
HR
Sales

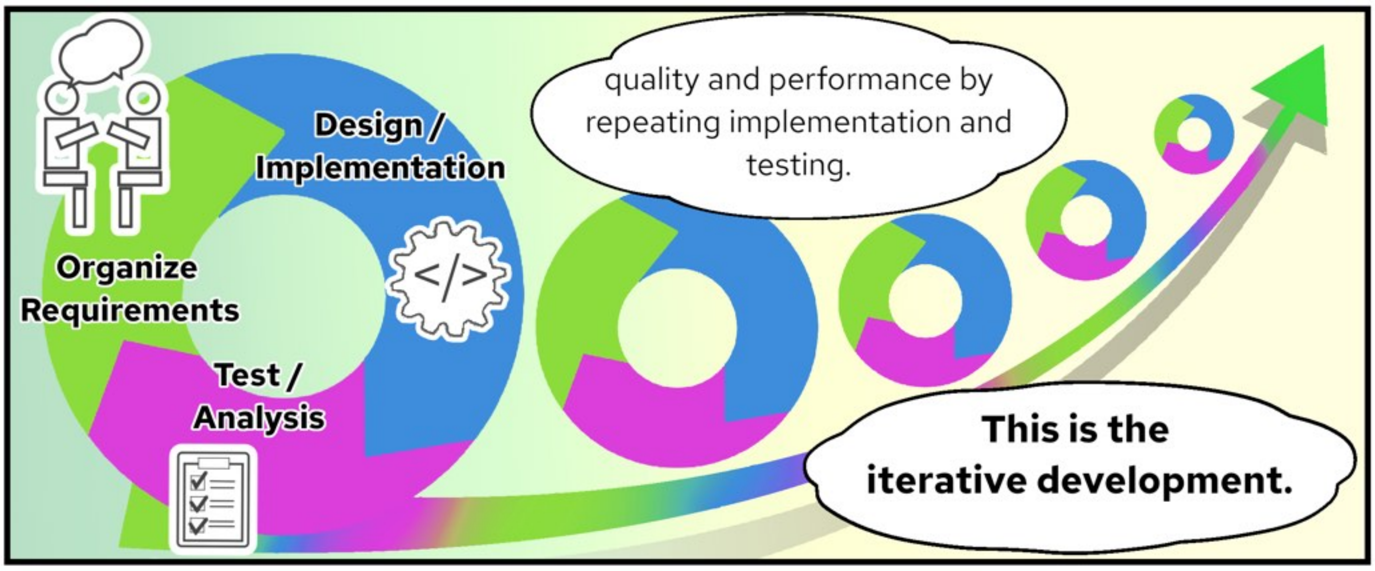

Let's put aside messy and garbage source codes currently remain.
Who knows best about business rules?
ystem A
SystemB
SystemC
SystemD


Even in similar tasks, rules may be different by different person in charge.
If the rules are OK to be different, we should implement as different rule.
Order
Purchase Cost
SG&A Expense
But If they should be the same, then make them unified.
HR DB
The important thing when you ask them is
to organize with business terminology by business unit.
Bonus
Allowance
Severance Pay
Transfer history
Evaluation
Training
Social Insurance
Personal Account
Adjustment
But would it take so much time as same as redefining requirements from scratch?
As a merit of Rule Driven Development,
you can start to organize what business rule should be, as starting point.
Purchase FI
Sales Process
To do that, it is important to use rule engine and
proceed with iterative development.
Iterative development by using rule engine...
It takes much shorter time and more efficient than discovering rules you need from a garbage source code.

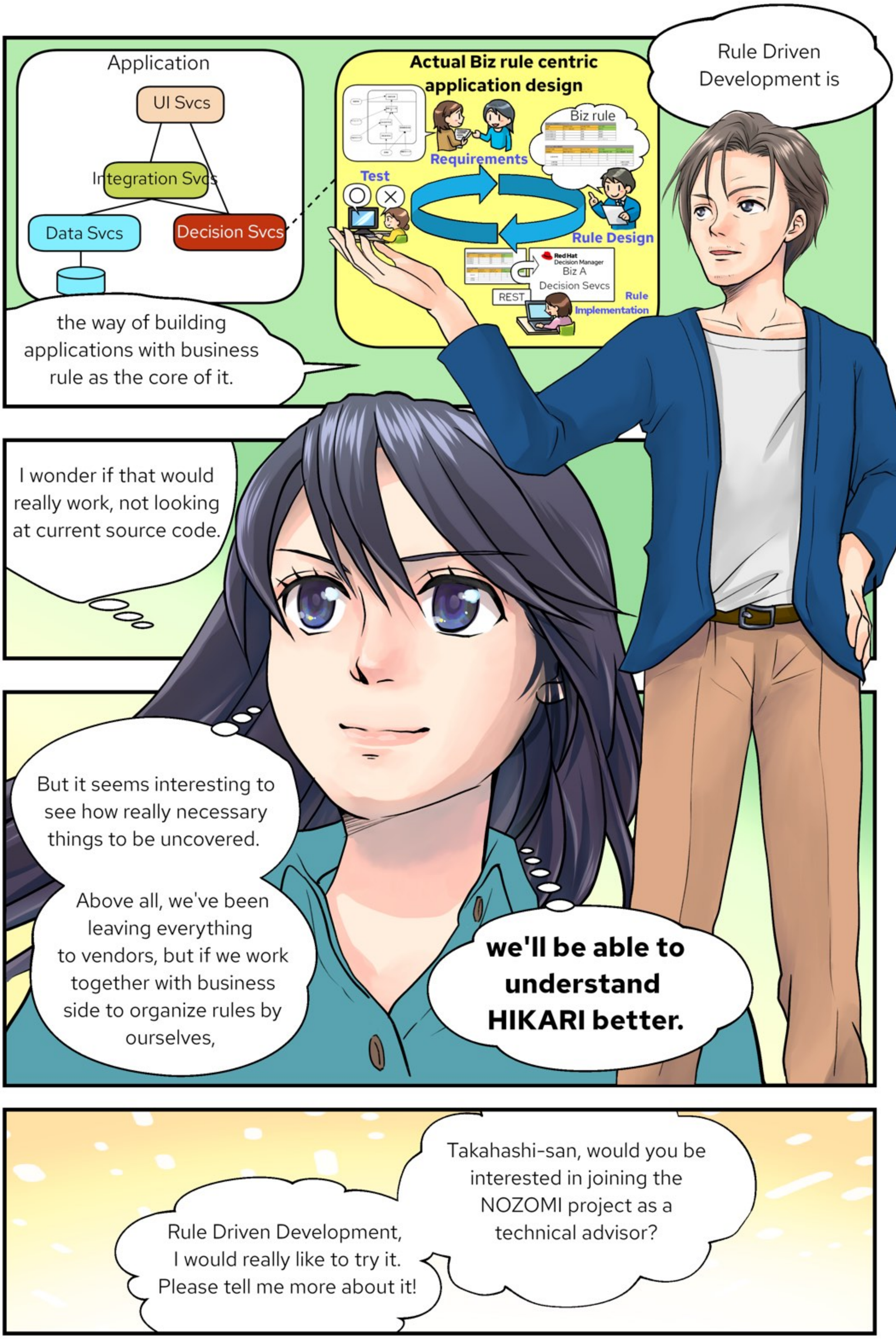

Red Hat Decision Manager
A rule engine is a mechanism to intuitively execute a rule such as "if A, then B", and to make it work efficiently.
Evaluate CONDITION
Do ACTION
By organizing business rules in this way, you can always check any discrepancies in recognition,

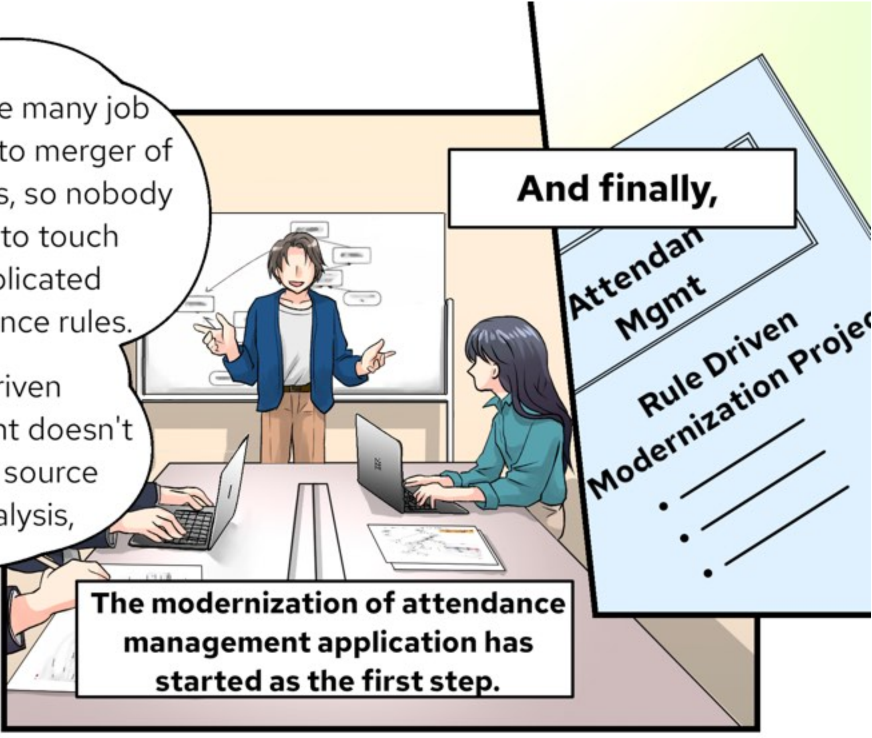
RuleTable "Campaign point rate"

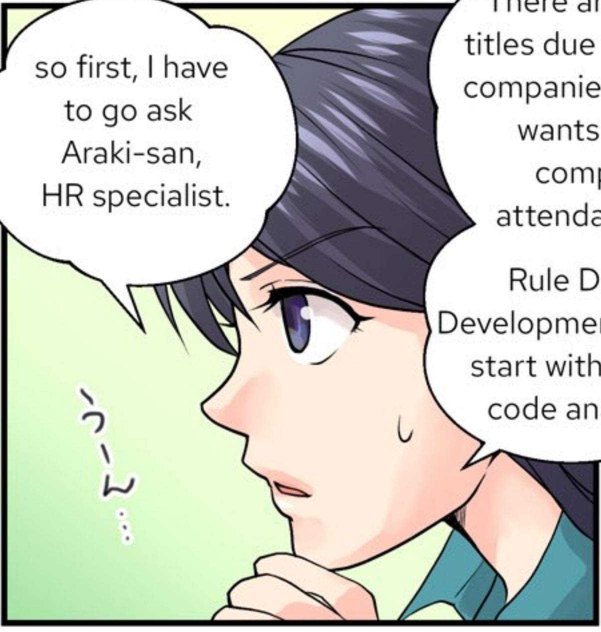
	CONDITION				ACTION
Name of Camp	Date		Amount		Point rate
Enjoy your culture?	3rd Nov.	23rd Nov.		$49.99	0%
			$50.00	$99.99	10%
			$100.00	$399.99	20%
Black Friday	24th Nov.	30th Nov.		$39.99	0%
			$40.00	$199.99	20%


Design / Implementation
Organize Requirements
Test / Analysis
quality and performance by repeating implementation and testing.
This is the iterative development.

Organize requirements, implementation, and testing... repeat this process?
It looks safe and fast.
like this...

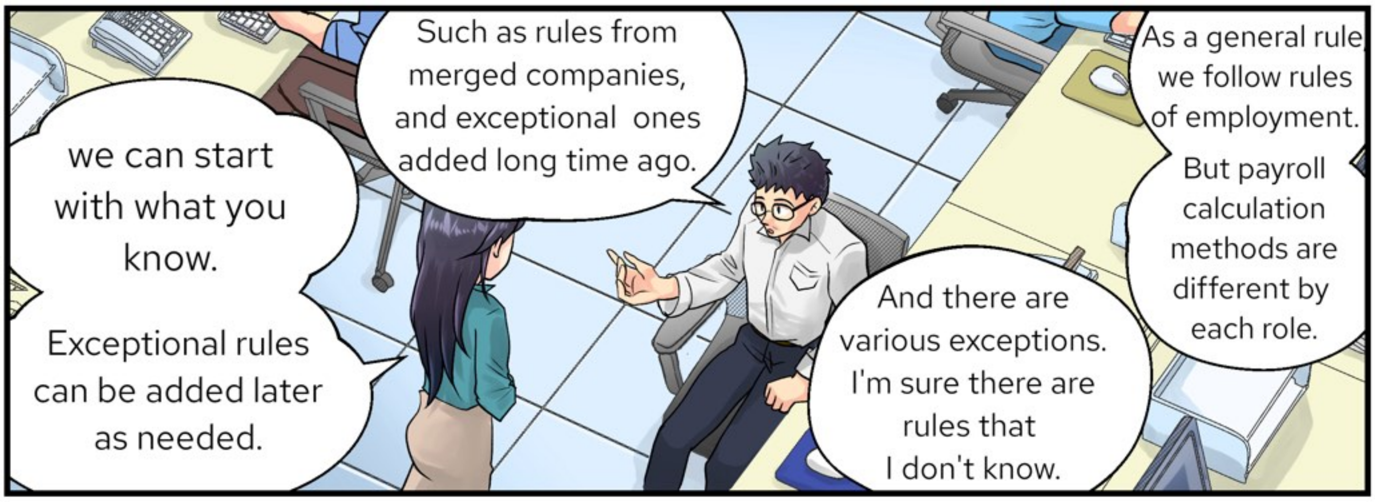

Rule Driven Development is
Application
UI Svcs
Integration Svcs
Data Svcs
Decision Svcs
Actual Biz rule centric application design
Requirements
Biz rule
Test
Rule Design
Red Hat Decision Manager
Biz A
Decision Sevcs
REST
Rule Implementation
the way of building applications with business rule as the core of it.
I wonder if that would really work, not looking at current source code.
But it seems interesting to see how really necessary things to be uncovered.
Above all, we've been leaving everything to vendors, but if we work together with business side to organize rules by ourselves,
we'll be able to understand HIKARI better.
Takahashi-san, would you be interested in joining the NOZOMI project as a technical advisor?
Rule Driven Development, I would really like to try it. Please tell me more about it!

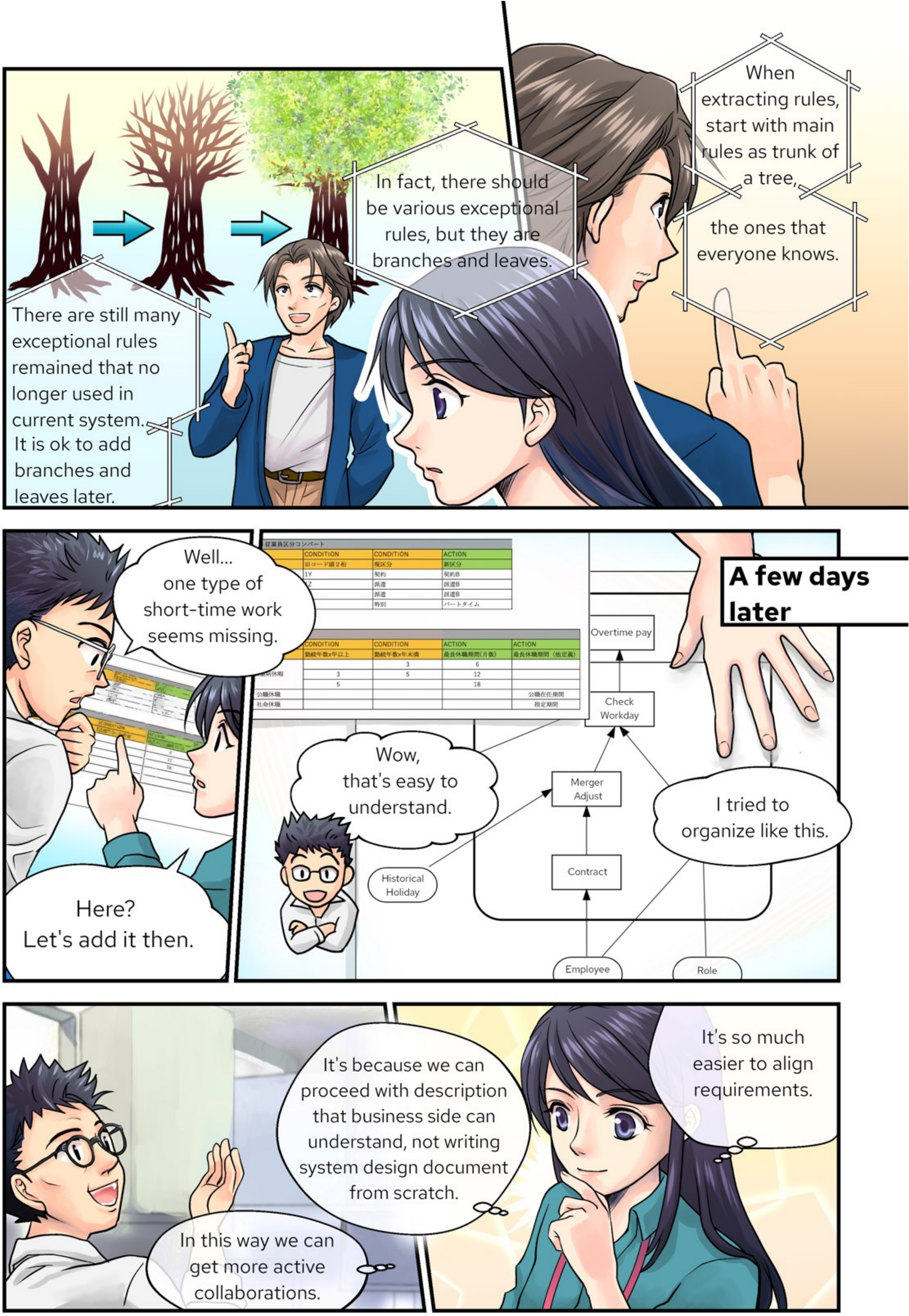

And finally,
Attendan Mgmt
Rule Driven Modernization Projec
The modernization of attendance management application has started as the first step.

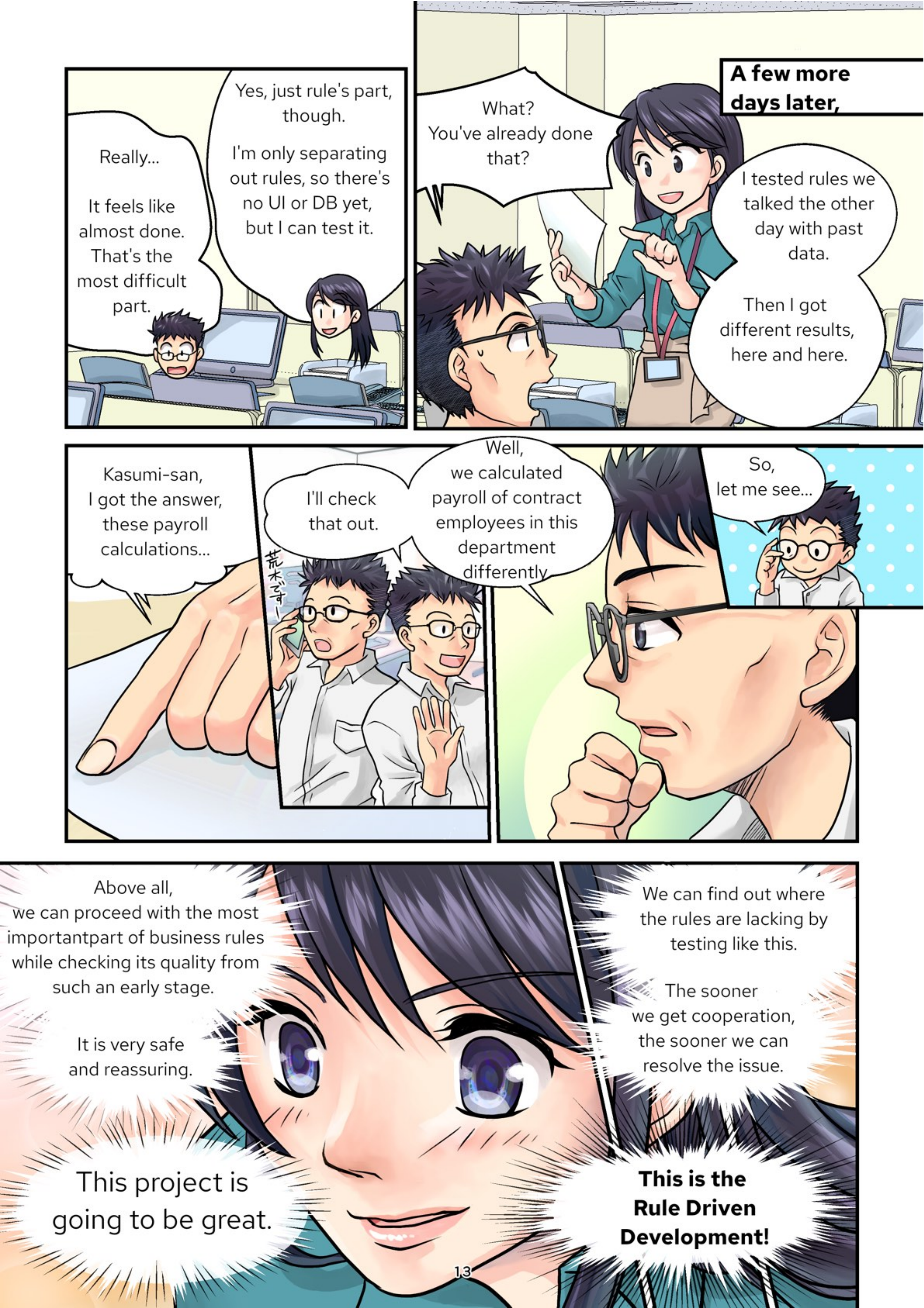

There are many job titles due to merger of companies, so nobody wants to touch complicated attendance rules.
Rule Driven Development doesn't start with source code analysis,
so first, I have to go ask Araki-san, HR specialist.
う〜ん…

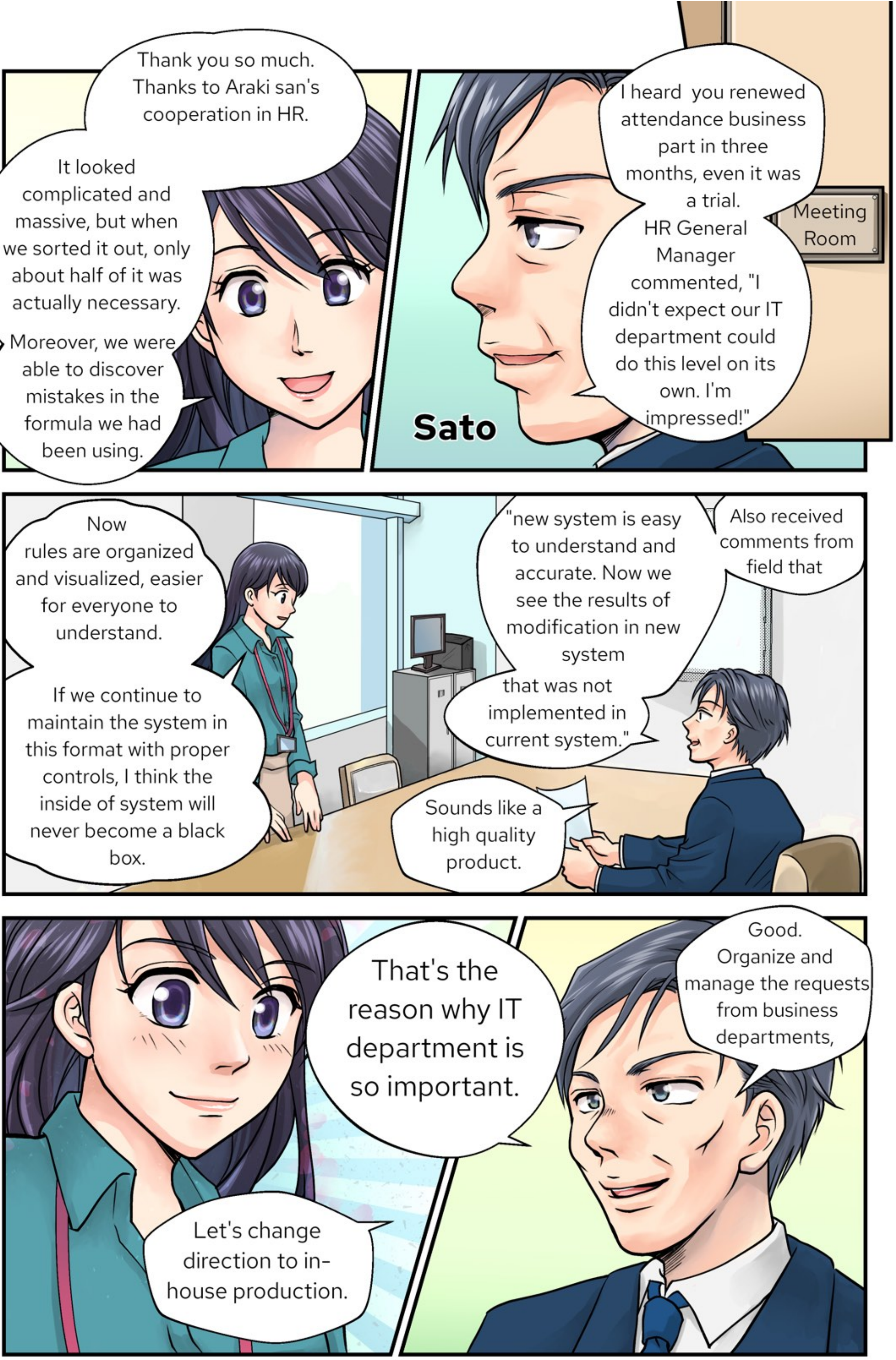
Sorry to bother you, but I really need your help...
Yes

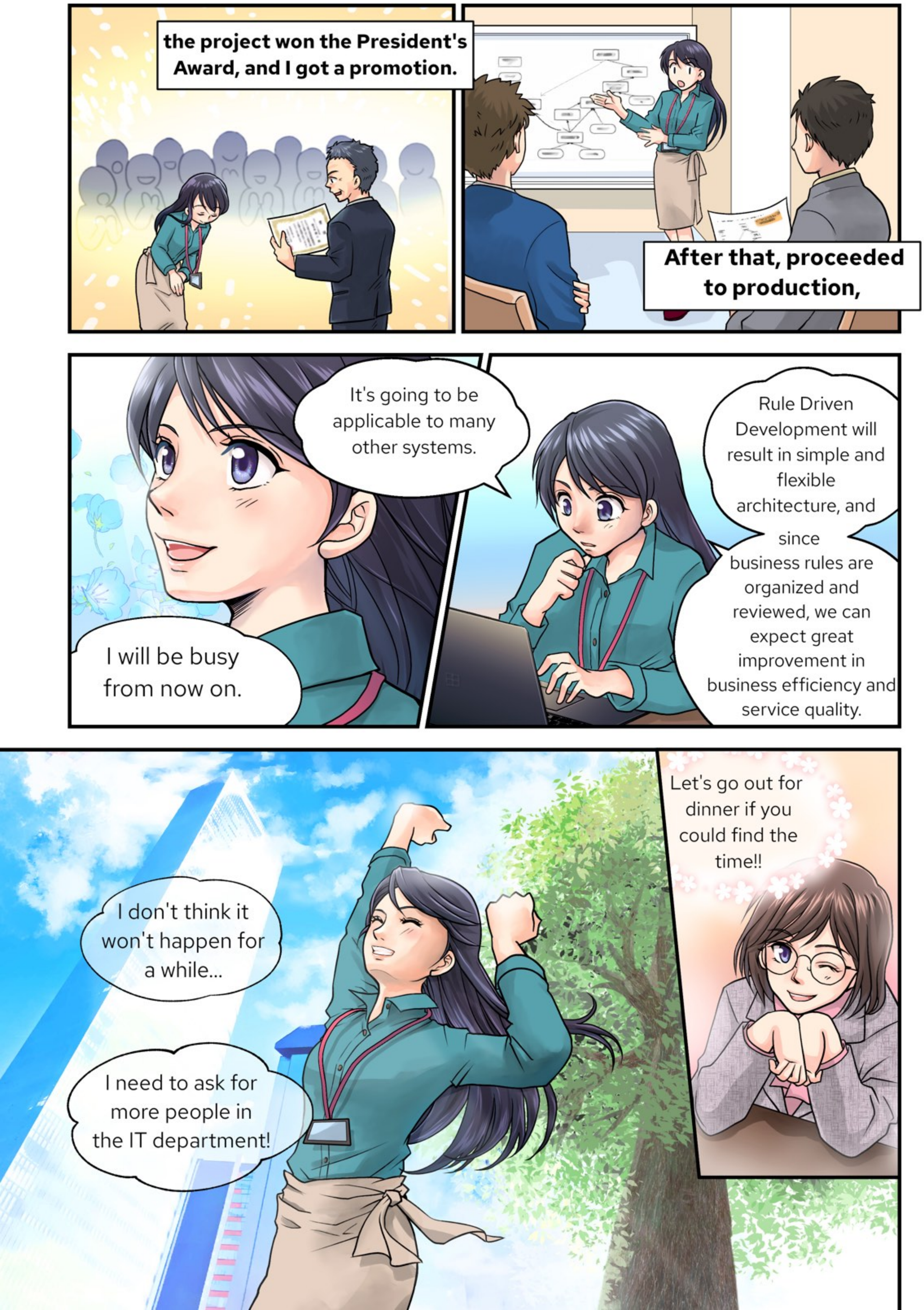
I see, so you want me to tell you about the attendance rules.
Araki


As a general rule, we follow rules of employment.
But payroll calculation methods are different by each role.
And there are various exceptions. I'm sure there are rules that I don't know.
Such as rules from merged companies, and exceptional ones added long time ago.
we can start with what you know.
Exceptional rules can be added later as needed.


When extracting rules, start with main rules as trunk of a tree,
the ones that everyone knows.
In fact, there should be various exceptional rules, but they are branches and leaves.
There are still many exceptional rules remained that no longer used in current system. It is ok to add branches and leaves later.
Well... one type of short-time work seems missing.
Here? Let's add it then.
A few days later
I tried to organize like this.
Wow, that's easy to understand.
Overtime pay
Check Workday
Merger Adjust
Contract
Historical Holiday
Employee
Role
CONDITION
ACTION
It's so much easier to align requirements.
It's because we can proceed with description that business side can understand, not writing system design document from scratch.
In this way we can get more active collaborations.


Really... It feels like almost done. That's the most difficult part.
Yes, just rule's part, though.
I'm only separating out rules, so there's no UI or DB yet, but I can test it.
A few more days later,
What? You've already done that?
I tested rules we talked the other day with past data.
Then I got different results, here and here.
Kasumi-san, I got the answer, these payroll calculations...
I'll check that out.
荒木です
Well, we calculated payroll of contract employees in this department differently
So, let me see...
Above all, we can proceed with the most importantpart of business rules while checking its quality from such an early stage.
It is very safe and reassuring.
We can find out where the rules are lacking by testing like this.
The sooner we get cooperation, the sooner we can resolve the issue.
This project is going to be great.
This is the Rule Driven Development!


Thank you so much. Thanks to Araki san's cooperation in HR.
It looked complicated and massive, but when we sorted it out, only about half of it was actually necessary.
Moreover, we were able to discover mistakes in the formula we had been using.
Sato
I heard you renewed attendance business part in three months, even it was a trial.
HR General Manager commented, "I didn't expect our IT department could do this level on its own. I'm impressed!"
Meeting Room
Also received comments from field that
"new system is easy to understand and accurate. Now we see the results of modification in new system
that was not implemented in current system."
Now rules are organized and visualized, easier for everyone to understand.
If we continue to maintain the system in this format with proper controls, I think the inside of system will never become a black box.
Sounds like a high quality product.
That's the reason why IT department is so important.
Let's change direction to in-house production.
Good. Organize and manage the requests from business departments,


After that, proceeded to production,
the project won the President's Award, and I got a promotion.
Rule Driven Development will result in simple and flexible architecture, and
since business rules are organized and reviewed, we can expect great improvement in business efficiency and service quality.
It's going to be applicable to many other systems.
I will be busy from now on.
Let's go out for dinner if you could find the time!!
I don't think it won't happen for a while...
I need to ask for more people in the IT department!

Key products used in modernization provided by Red Hat

The rule engine needed for modernization successfully!

Red Hat
Decision Manager *

Red Hat
Consulting

We'll Help you with Rule Driven Development!

Red Hat
Process Automation Manager *

For task and progress management

The definitive to integreate system and services

Red Hat
Fuse

Essential Items for Microservices

Red Hat
AMQ

How about new API businesses?

Red Hat
3scale API Management

Red Hat
OpenShift Container Platform

This is the container env.!

Red Hat
Data Grid

In Memory DB for ultra-high speed processing

Red Hat
Ansible Automation Platform

Let's modernize operations to make it more intelligent.

All picture drawed by @koyagi_rm : Special Thanks!!

Please contact Red Hat! We can help you!!

* Red Hat is transferring the Process Automation Manager and Decision Manager products to IBM. Red Hat PAM/DM V7.13 was released on August 4, 2022, and will be the last product version released by Red Hat . Future product versions, beginning with PAM and DM V8.0, are available from IBM. IBM PAM/DM 8.0 was released on July 22, 2022.
https://access.redhat.com/articles/6968468
https://community.ibm.com/community/user/automation/viewdocument/faq-ibm-process-automation-manage