



BTH

SERL Sweden
LEADING SOFTWARE ENGINEERING



AI and LLMs in Software Engineering

Hosted by:

Tony Gorschek and Julian Frattini

BTH **SERL Sweden**
LEADING SOFTWARE ENGINEERING

S.E.R.T
reThink • reThought

"AI" and the Virtual Pandoras Box of Sweet and Sour

SERL

*Software Engineering Research Lab
@ Blekinge Institute of Technology, Sweden*



www.rethought.se
www.promisedu.se

who is



→ Engineer (first) / Problem Solver / Researcher

- Professor, PhD (Tekn. Dr.) Software Engineering, M.Sc. Computer Science + B.Sc. Economy
- 14 years in industry (6 start-ups, CTO, Senior Executive Consultant, Chief Architect, Technical Advisor, Developer, Product Manager/Business Analyst, Investor)
- 16 years in research (Technology Product Management, Requirements Engineering, Quality Assurance, Agile transformation/continuous development, Lean Product development, Value based product development, Applied AI)
- Research leader and project manager *S.E.R.T.*



MaxKompetens



Volvo Construction Equipment

HIRENCO



ERICSSON



Spotify

Swedbank



SONY

TimePeopleGroup



TOLPAGORNI
product management

Knowledge Foundation



FORTNOX

Småföretagens bästa vän

S.E.R.T.
reThink • reThought

Synteda



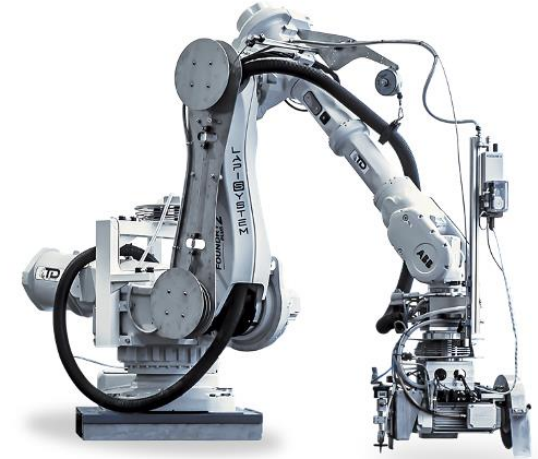
itestra
be excellent

Handelsbanken

Mobile Financial Services

fortiss

BTH SERL Sweden
LEADING SOFTWARE ENGINEERING



AI for SIPS engineering

AI4SE

We use AI as a tool to engineer SIPS (Software Intensive Products and Services) – e.g. developers use an AI co-pilot to code, a requirements engineer uses it to analyze requirements...

Engineering for AI based SIPS

SE4AI

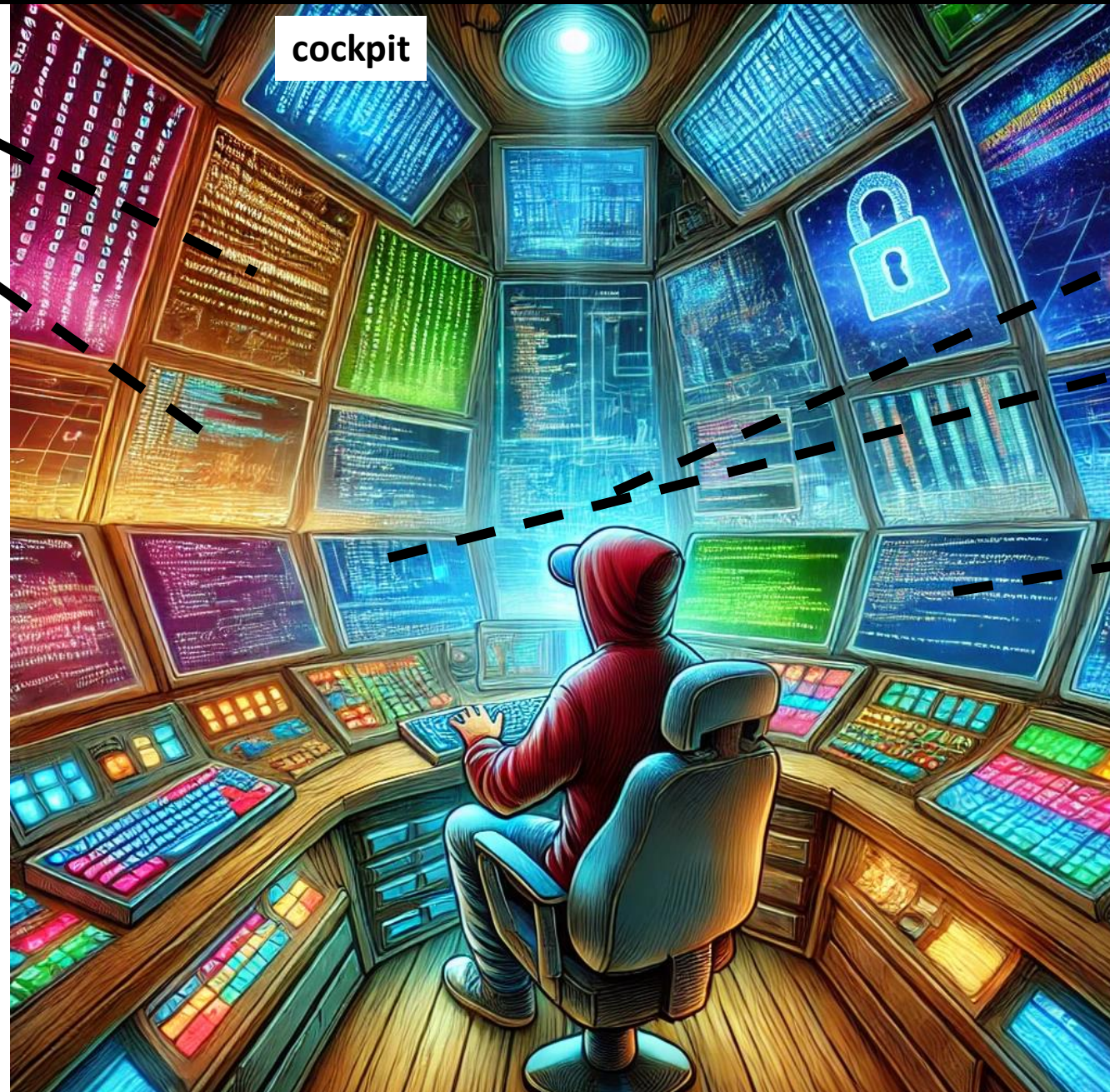
When we develop SIPS, it has an “AI component” included as a part of it (developed, bought..) that we have to put requirements ON and architect, design, develop and validate...

... so what have we seen so far...

...perspective



...perspective (what can we do (well-ish) today)



cockpit

Pre-study/investigate area

Requirements Engineering

Create reqs

Fast mock-ups

Design / Modeling

Coding

Test case generation

Testing

Specialist LLMs (Agents)
RAGs
Utilizing LLMs and tools...
Combine models...

... so what can we do today ...

...requirements engineering today...

Create a requirement specification in SAFe style that covers the sale instance in a POS system specifically in relation to adding a customer to the customer loyalty program

Main Flow:

1. Login to POS System

1. The salesperson enters their credentials (username and password).
2. The salesperson clicks the "Login" button.
3. The system verifies the credentials and grants access to the main dashboard.

2. Greet Customer and Start Sale...

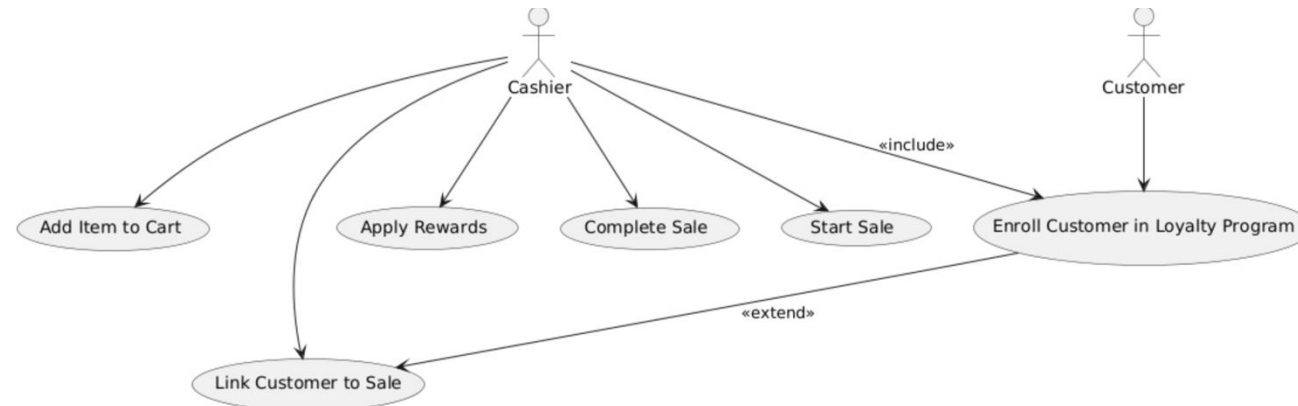
...

Subflows:

3a. Product Not Found

- If the product barcode is not found, the system displays an error message...

...



...requirements engineering today...

As a next step, write a compliance specification of these requirements

1.Global:

1. **ISO/IEC 27001**: Information security management standard.
2. **Payment Card Industry Data Security Standard (PCI-DSS)**: Ensures...

2.European Union:

1. **General Data Protection Regulation (GDPR)**: Ensures privacy and protection of personal data.

3.United States:

1. **California Consumer Privacy Act (CCPA)**: Data privacy regulation for California residents.
2. **Health Insurance Portability and Accountability Act (HIPAA)**: ...

...

4.Japan:

1. **Act on the Protection of Personal Information (APPI)**....

...

5.Germany:

1. **Bundesdatenschutzgesetz (BDSG)**: ...

...

6.Accessibility (Applicable Globally but with Regional Mandates):

1. **Web Content Accessibility Guidelines (WCAG 2.1)** and **Americans with Disabilities Act (ADA)**

...requirements engineering today...

Break down how you could use GQM to metric level to measure the usability of this requirement for the employee using the POS

GQM Approach to Measure Usability for the Employee Using the POS System:

Goal: Analyze the usability of the POS system for employees (cashiers) during customer loyalty program enrollment to improve ease of use and efficiency.

Questions:

1.Efficiency:

1. How quickly can cashiers complete the loyalty program enrollment during checkout?
2. How many steps ...

2.Effectiveness:

1. How accurately are cashiers able to enroll customers without errors?
2. How often...

...

Metrics:

1.Efficiency Metrics:

1. Average time taken to enroll a customer into the loyalty program (in seconds).
2. Number of steps required to complete the enrollment process.

2.Effectiveness Metrics:

1. Error rate during the ...

...

Write a manual test case for these requirements next

Test Case ID: TC-001

Title: Customer Enrollment into Loyalty Program via POS System

Objective: Verify that the POS system allows cashiers to enroll customers into the loyalty program during checkout in compliance with GDPR.

Preconditions:

- The POS system is up and running.
- The cashier is logged into the POS system.

Test Steps:

- 1.Start a new transaction in the POS system.
- 2.During the checkout process, verify if the POS prompts the cashier to ask the customer if they want to join the loyalty program.
- 3....
- ...

Expected Results:

- The POS system prompts the cashier to offer loyalty program enrollment if the customer is not already enrolled.

Postconditions:

- The customer is enrolled in the loyalty program, and their information is stored securely

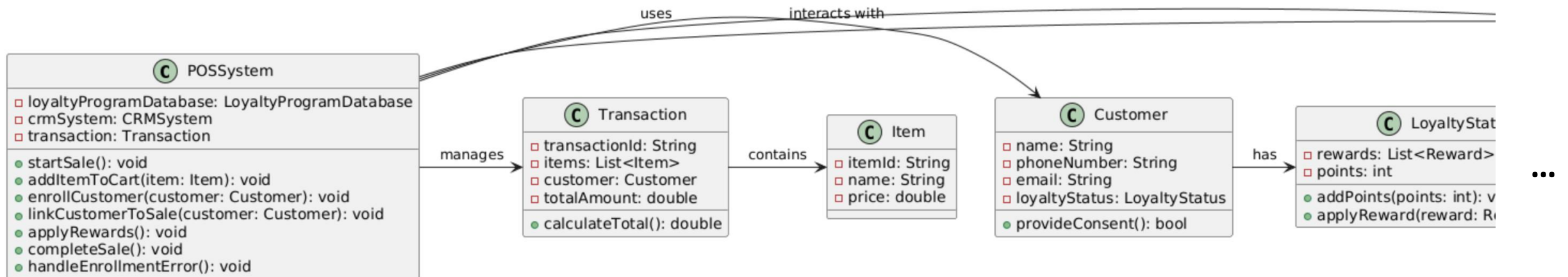
...

Write an automated test case

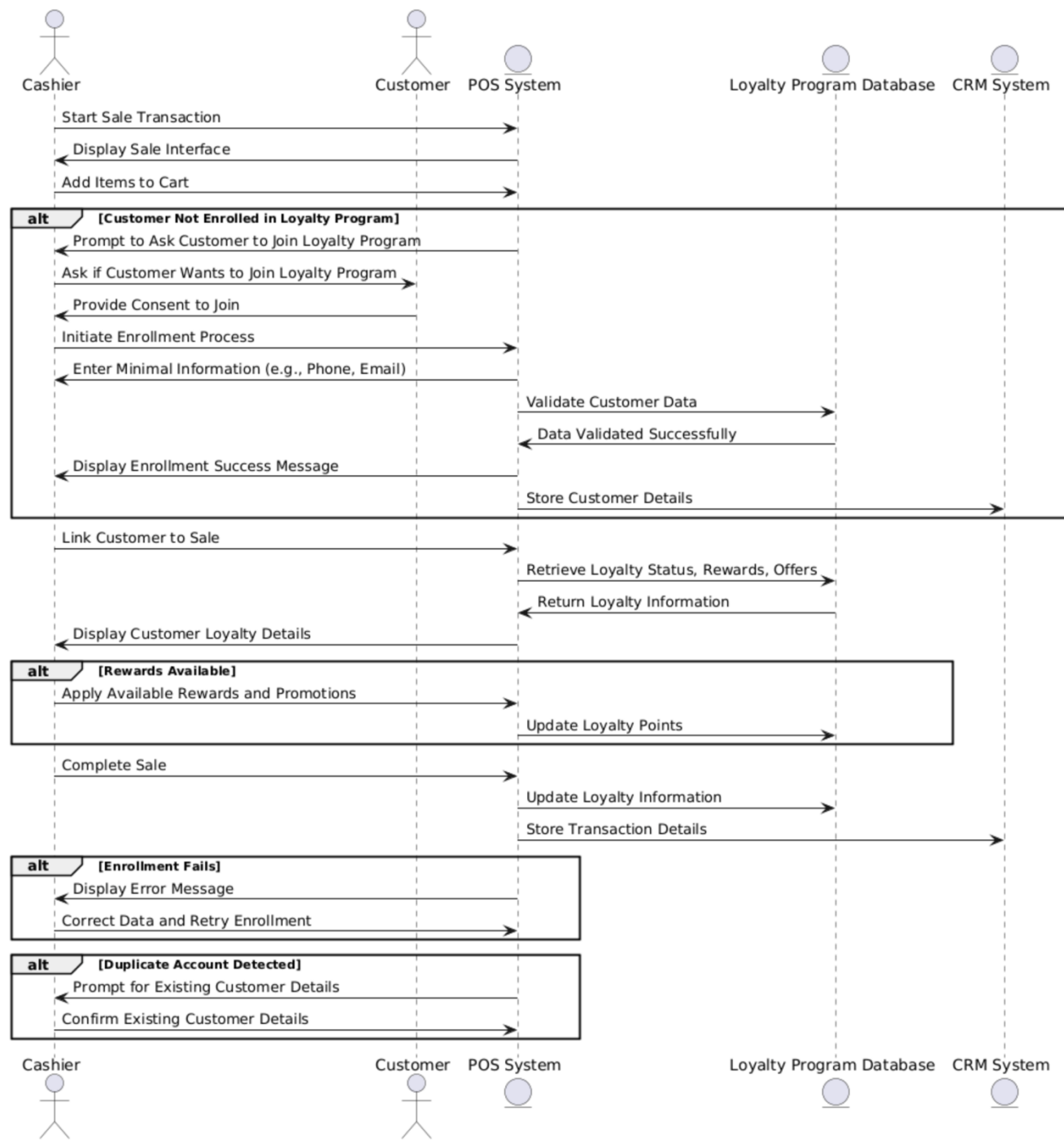
```
1 import unittest
2 from pos_system import POSSystem, LoyaltyProgramDatabase, CRMSystem, Customer, Item
3
4 class TestPOSSystem(unittest.TestCase):
5
6     def setUp(self):
7         # Setting up instances for testing
8         # Create instances of the loyalty program database and CRM system
9         self.loyalty_program_db = LoyaltyProgramDatabase()
10        self.crm_system = CRMSystem()
11        # Create an instance of the POS system with the loyalty program and CRM system
12        self.pos_system = POSSystem(self.loyalty_program_db, self.crm_system)
13
14        # Example customer and items for testing
15        self.customer = Customer("John Doe", "1234567890", "john.doe@example.com") # Create a customer instance
16        self.item1 = Item("ITEM001", "Milk", 2.5) # Create an item instance for Milk
17        self.item2 = Item("ITEM002", "Bread", 1.5) # Create an item instance for Bread
18
19    def test_start_sale(self):
20        # Test case TC-001: Start Sale Transaction
21        # Verify that a new sale transaction can be started
22        self.pos_system.start_sale() # Start a new sale transaction
23        self.assertIsNotNone(self.pos_system.transaction, "Transaction should be started.") # Check that a transaction is active
24
```

...

Create an UML class diagram for the POS



Create a sequence UML diagram for the POS



...code (support) today...

Write the code
please

```
1  class POSSystem:
2      def __init__(self, loyalty_program_db, crm_system):
3          self.loyalty_program_db = loyalty_program_db
4          self.crm_system = crm_system
5          self.transaction = None
6
7      def start_sale(self):
8          self.transaction = Transaction()
9          print("Sale transaction started.")
10
11     def add_item_to_cart(self, item):
12         if self.transaction:
13             self.transaction.add_item(item)
14             print(f"Item {item.name} added to cart.")
15         else:
16             print("No active transaction. Please start a sale first.")
17
18     def enroll_customer(self, customer):
19         if customer.provide_consent():
20             if self.loyalty_program_db.validate_customer_data(customer):
21                 self.crm_system.store_customer_details(customer)
22                 print(f"Customer {customer.name} enrolled in loyalty program.")
23             else:
24                 self.handle_enrollment_error()
25         else:
26             print("Customer did not provide consent for enrollment.")
27
28     def link_customer_to_sale(self, customer):
29         if self.transaction:
30             self.transaction.customer = customer
31             loyalty_status = self.loyalty_program_db.retrieve_loyalty_status(customer.phone_number)
32             customer.loyalty_status = loyalty_status
33             print(f"Customer {customer.name} linked to sale. Loyalty status retrieved.")
34         else:
35             print("No active transaction to link customer to.")
36
```

...code (support) today...

Write the code
please

```
1  class POSSystem:
2      def __init__(self, loyalty_program_db, crm_system):
3          self.loyalty_program_db = loyalty_program_db
4          self.crm_system = crm_system
5          self.transaction = None
6
7      def start_sale(self):
8          self.transaction = Transaction()
9          print("Sale transaction started.")
10
11     def add_item_to_cart(self, item):
12         if self.transaction:
13             self.transaction.add_item(item)
14             print(f"Item {item.name} added to cart.")
15         else:
16             print("No active transaction. Please start a sale first.")
17
18     def enroll_customer(self, customer):
19         if customer.provide_consent():
20             if self.loyalty_program_db.validate_customer_data(customer):
21                 self.crm_system.store_customer_details(customer)
22                 print(f"Customer {customer.name} enrolled in loyalty program.")
23             else:
24                 self.handle_enrollment_error()
25         else:
26             print("Customer did not provide consent for enrollment.")
27
28     def link_customer_to_sale(self, customer):
29         if self.transaction:
30             self.transaction.customer = customer
31             loyalty_status = self.loyalty_program_db.retrieve_loyalty_status(customer.phone_number)
32             customer.loyalty_status = loyalty_status
33             print(f"Customer {customer.name} linked to sale. Loyalty status retrieved.")
34         else:
35             print("No active transaction to link customer to.")
36
```

...code (support) today...

Add comments for
each line

(oh, with comments...)

```
1  class POSSystem:
2      # Constructor to initialize the POS system with loyalty program and CRM systems
3  def __init__(self, loyalty_program_db, crm_system):
4      self.loyalty_program_db = loyalty_program_db # Loyalty program database instance
5      self.crm_system = crm_system # CRM system instance
6      self.transaction = None # Current transaction, initially None
7
8      # Method to start a new sale transaction
9  def start_sale(self):
10     self.transaction = Transaction() # Create a new transaction instance
11     print("Sale transaction started.") # Inform the cashier that the sale has started
12
13     # Method to add an item to the cart
14  def add_item_to_cart(self, item):
15     if self.transaction: # Check if a transaction is active
16         self.transaction.add_item(item) # Add the item to the transaction
17         print(f"Item {item.name} added to cart.") # Inform the cashier that the item was added
18     else:
19         print("No active transaction. Please start a sale first.") # Inform the cashier if no transaction is active
20
21     # Method to enroll a customer in the loyalty program
22  def enroll_customer(self, customer):
23     if customer.provide_consent(): # Check if the customer provides consent
24         if self.loyalty_program_db.validate_customer_data(customer): # Validate customer data
25             self.crm_system.store_customer_details(customer) # Store customer details in the CRM system
26             print(f"Customer {customer.name} enrolled in loyalty program.") # Inform the cashier of successful enrollment
27         else:
28             self.handle_enrollment_error() # Handle enrollment errors
29     else:
30         print("Customer did not provide consent for enrollment.") # Inform the cashier if consent is not given
31
32     # Method to link a customer to the current sale transaction
33  def link_customer_to_sale(self, customer):
34     if self.transaction: # Check if a transaction is active
35         self.transaction.customer = customer # Link the customer to the transaction
36         loyalty_status = self.loyalty_program_db.retrieve_loyalty_status(customer.phone_number) # Retrieve loyalty status
37         customer.loyalty_status = loyalty_status # Update customer's loyalty status
38         print(f"Customer {customer.name} linked to sale. Loyalty status retrieved.") # Inform the cashier
39     else:
40         print("No active transaction to link customer to.") # Inform the cashier if no transaction is active
```

... so does it work ...

(well... kind-of-ish...)

does it work...

REQUIREMENTS ENGINEERING

Pre-study stuff good start...
Very fast creation of base parts
Reasonably complete and accurate
False sense for completeness
Very good to do stuff you normally ignore...

Efficiency: +++
Effectiveness: +

TESTing

Create a fast base of cases
Better completion than expected...
Automation capabilities so – so...
Better manual test cases than many people write...

Efficiency: ++
Effectiveness: +/-

DESIGN related

Not bad... fast track UML diagrams rather reasonable
Not really optimized in terms of big picture (architecture)...
Manual work significant for anything non-trivial

Efficiency: +
Effectiveness: (+)/-

CODING

Better than expected...
Significant head start
Significant clean up, and especially iterating with changes in design...
Interesting to see that promising results if design is changed code generation is often improved...

Efficiency: ++
Effectiveness: "meh"

OTHER

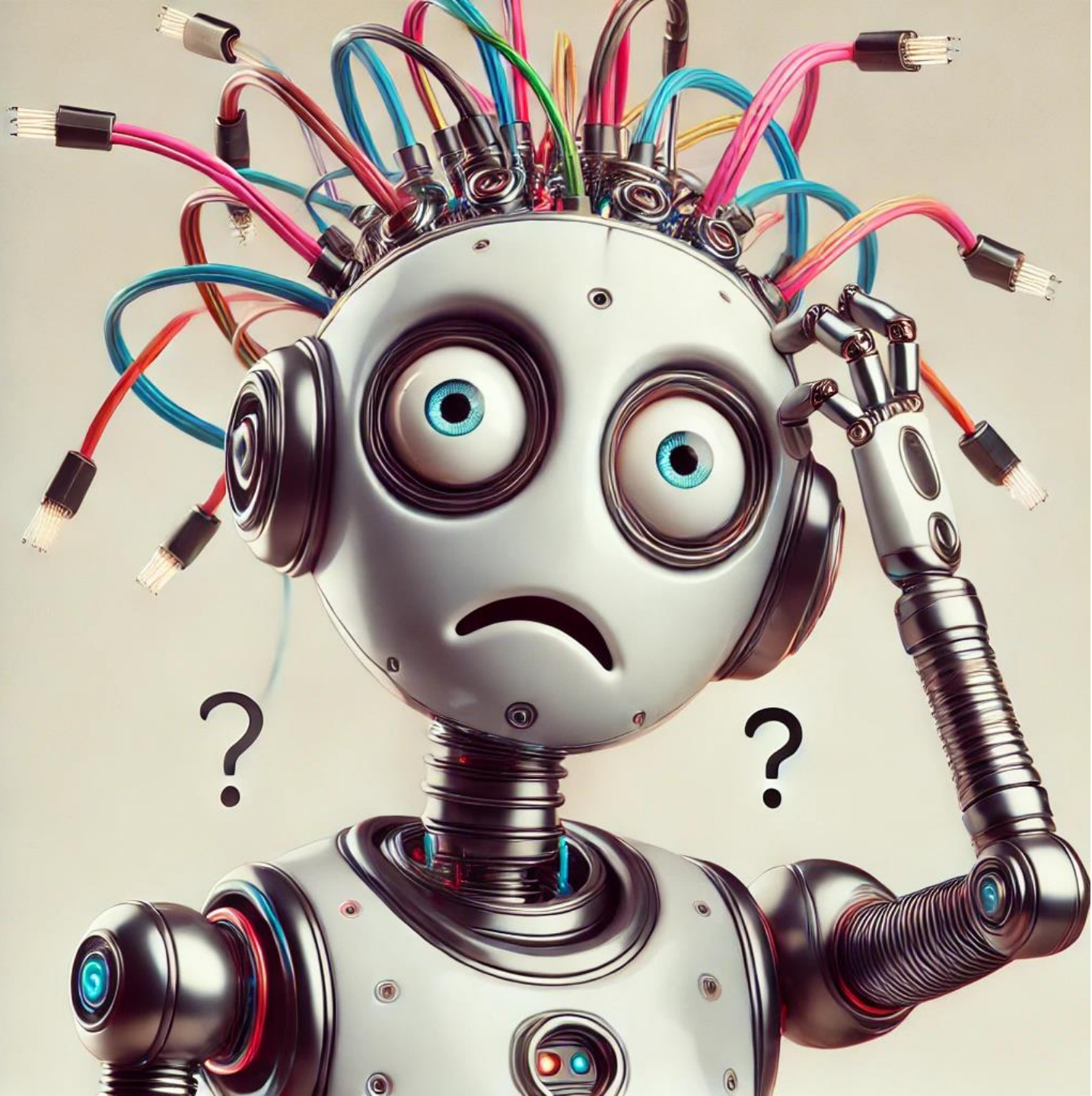
Good summary of compliance (standards.. etc.)
Good start to a STRIDE analysis...
...
I create a lot of fun pictures with AI to add to my powerpoints with no real reason...

Efficiency: ++
Effectiveness: +

... so now what ...

(future and predictions n stuff...)

Well I don't know...
... but lets guess...



fundamentals one: “how do we know”

“If we can judge the quality of the output, we per definition do not need the output...!”*

The opacity problem in the context of using “AI”

Lack of model understanding (transparency)

Unreliable confidence estimates

Bias detection

Opacity of the training data

Limited explainability

EFFICIENCY GAINS ARE FUNDAMENTAL (NOW) and will increase...

**“Tool integration” rather than stand alone
New ways to test “truth” and “bias” (3rd party?)
Guard rail transparency**

*not true as we can get output faster...

fundamentals two: “will it improve a lot in the future”

“the transformer models might not be significantly tweakable AND we can not just throw more data at it”

High cost(s) for training and training and training...

Guard rail race...

Make money... (lose money)

Limitations of the base models

Most data is already used for training

BUT!!! – Even if we have reached “PEAK LLM” – we have barely scratched the surface of application/use

More (smaller) specialized (efficient) models
Multimodal shift (combining e.g. text, vision...)
Domain specialized models

LLMS

OF LANGUAGE PERFORMANCE & CAPACITY

PERFORMANCE & CAPACITY

PERFORMANCE CAPACITY

SCALABILITY

SCALABILITY

PERFORMANCE

2.00
1.50
1.00
0.50
0.00



1.8.9%



GPT-3



GPT-4



CAPACITY



PERFORMANCE

DATA-3

MODEL SIZE

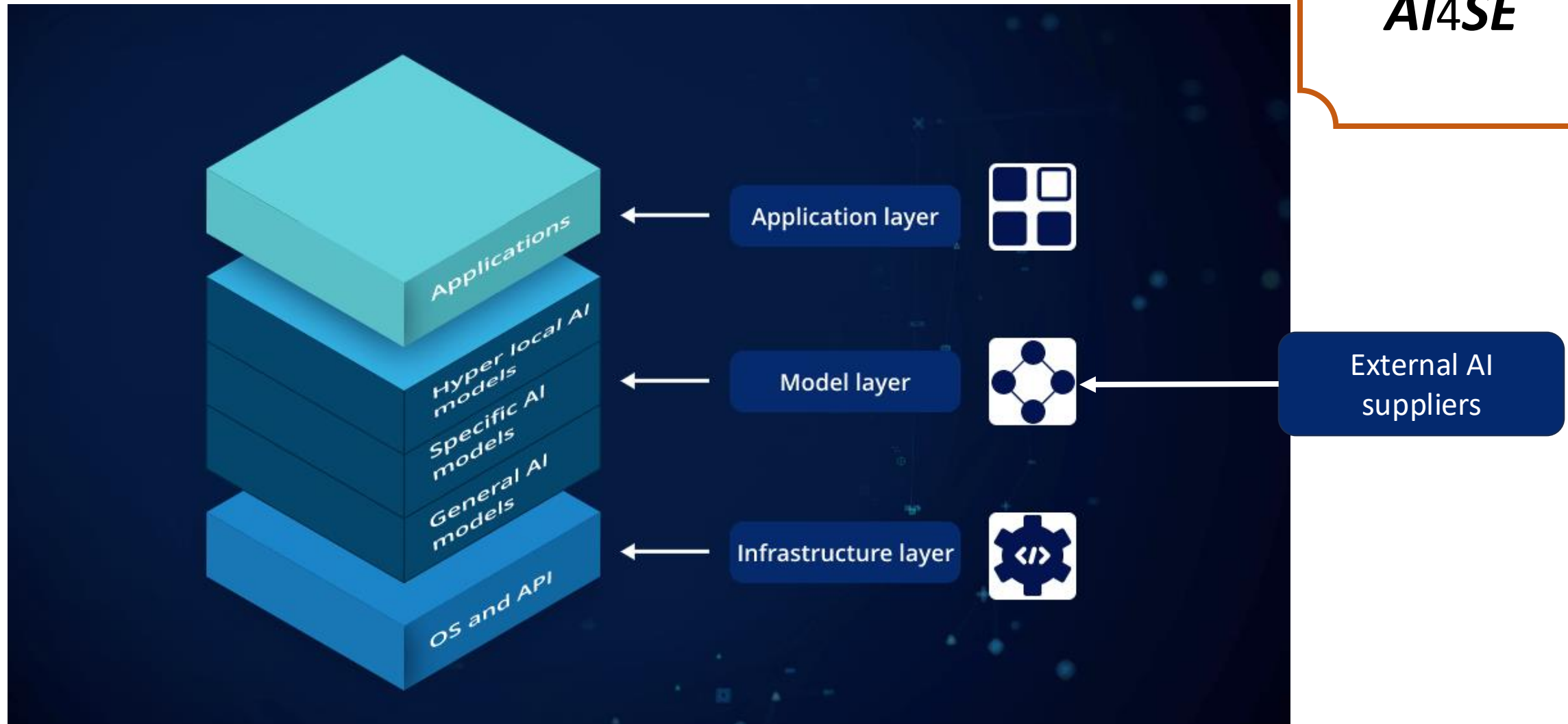
MODEL SIZE

MODEL SIZE

fundamentals three: “how do we test”

“how do we test a non-deterministic system?”

AI4SE

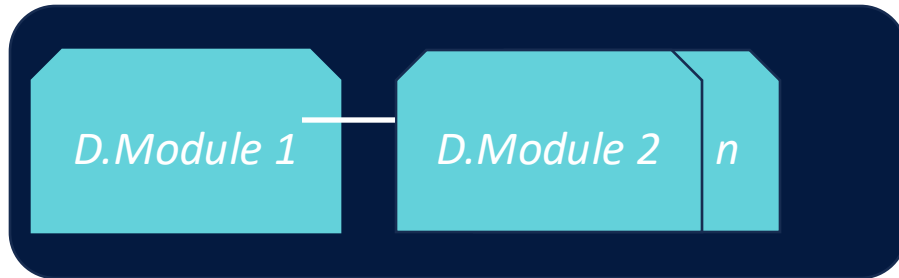


fundamentals three: “how do we test”

“how do we test a non-deterministic system?”

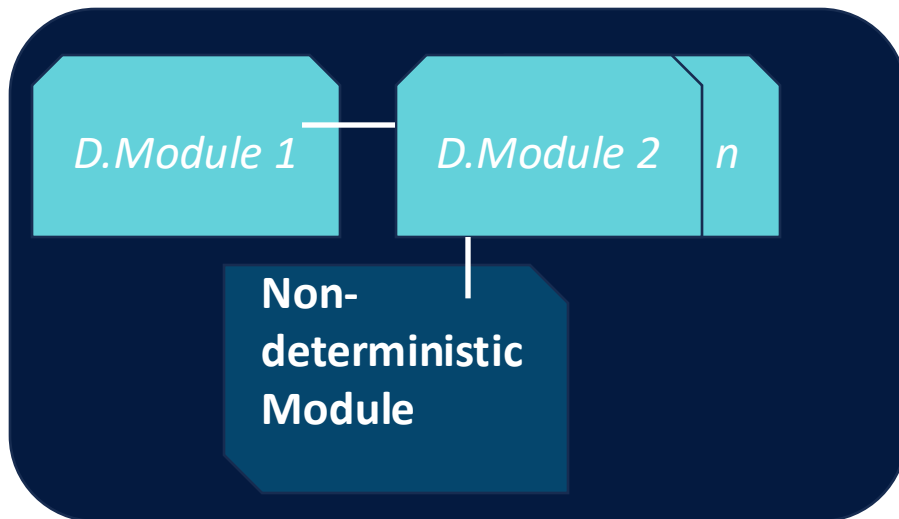
Traditional deterministic system...

(input)



(output) (predicable/testable output/behavior)

(input)



(output?)

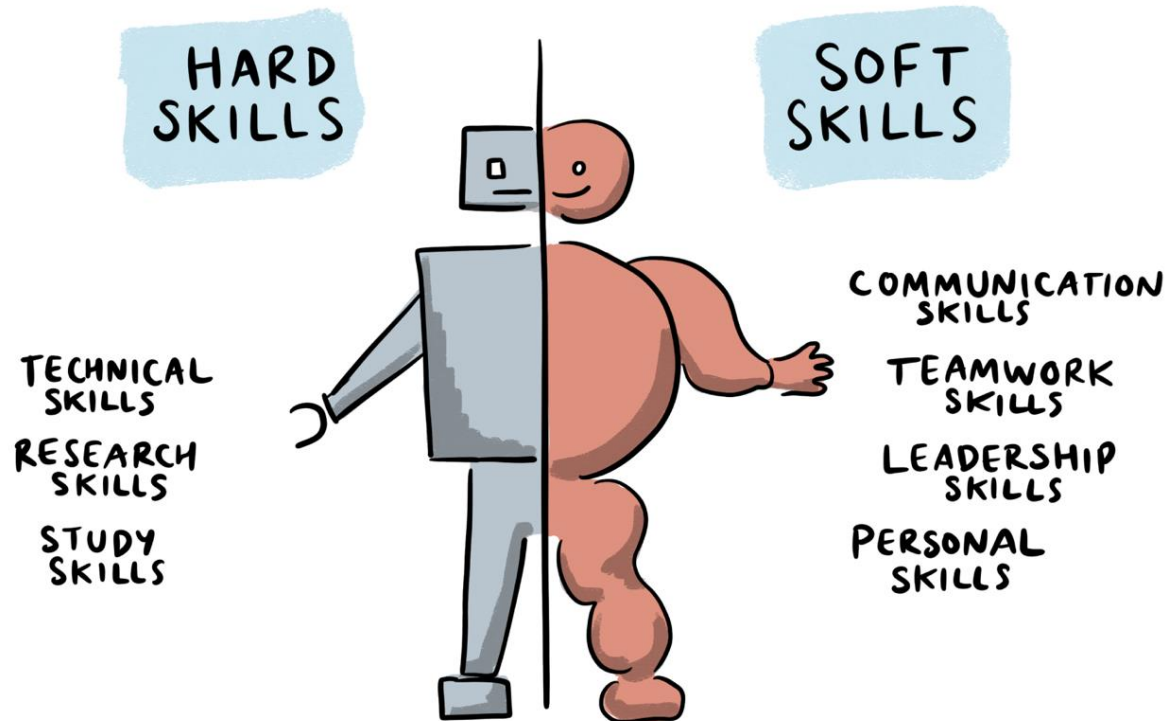
How do we “quality assure” systems with non-deterministic components (test)

- 3rd party (black/grey)
- Learning/under training in “real time”
- Complex interactions of multiple non-deterministic...

guesswork and thoughts...

Engineering

- **Better** programmers, but less programming (validation, what/how loop)
- More “documentation”, but **less** documenting (autogenerate JIT representations)
- “Rules” and “regulation” will be **counterproductive** (and slow things down)...
- **Architects** will still be important and have the big picture, but more **decision** makers
- **AI tool experts** linking multiple AIs and leveraging them in combination
- Work effectively in cross functional teams (communication, coordination, understand and be able to transfer the what/how/why loop)



guesswork and thoughts...

Role

- **Legal** as we know it now is “**in trouble**” – legal role and function will change...

Change

- Compliance will be much **easier** (process), Compliance will be much **harder** (outcome)

- Compliance will be the **great bottle-neck**...

- **Invention** (heavy human centric), **Innovation** (significantly AI driven)

New Roles

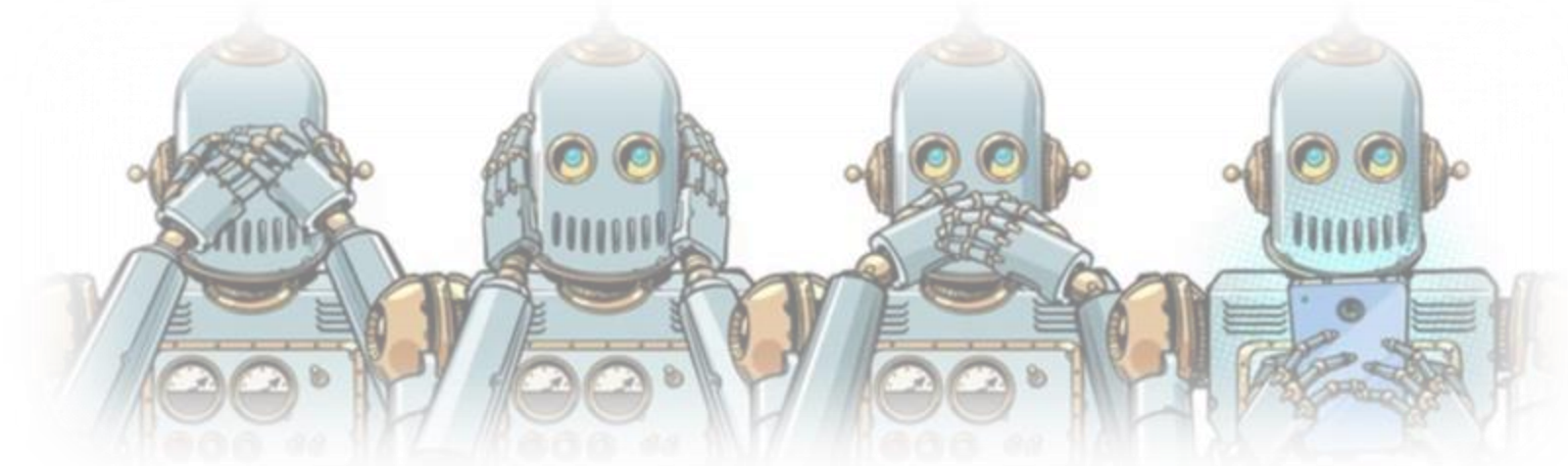
- **AI application experts and quality assurance (new roles, tools and expertise)**

- **New COST models... (combine architecture, business, vision...)**

- **CORE TECH EXPERTS (to write / use AIs to check AIs?!)**

- **Data scientists will be morphed into: “prompters” + “validators”**

...



work we are currently doing...

current ongoing research work... (examples)

Requirements engineering

Initial “collecting” requirements LLM based (current system(s), interviews, discussions, documentation, slides, old requirements etc.) → **Candidate SRS/Backlog** *in fraction of the time...* = Humans focus on WHY and REFINEMENT rather than “collection”

V&V

SIPS with non-deterministic components how will we “assure them”...

Tool

AI augmented tests (GUI, other...)

Tool integration of SDLC and “AI augmentation...”

Security

(Pre-) use of AI (LLMs...) how can we adapt current security models / standards to model threats...

Compliance

Compliance Automation how do we support compliance of actual outcome of a system? How do we monitor if changes in the system break compliance...?

Development

Compare AI assisted vs. Non-AI assisted programming... Productivity measures and benefits and challenges. Overall: speed **+**, quality **+/-**, long-term evolution/maintenance **-**

competence development...

PR.AI.SE
for and with AI

(20 company partners)
(5y, 20MSEK)
(Real university courses)
(Designed FOR professionals!)

Project 2025 in EDUCATION FOR PROFESSIONALS

(how do we re-train (rather enhance) the competence of professionals working today?!)

10+ brand
new courses
FOR
professionals

AI-enabled Product Management and Requirements Engineering

Software Architecture with AI and for AI-intensive SIPS

AI-based Software Modeling and Design

Programming with AI/LLMs

Software Testing and Quality Assurance with AI

Applied AI for Software Engineers

Software Evolution and Maintenance with AI

Security in AI-Driven SIPS

Ethics, Compliance and Legal Implications of AI in Software Engineering

Managing Software Development with AI: SAFe, DevOps, and CI/CD Integration

Interested?
(talk to me)

Thank you

BTH SERL Sweden
LEADING SOFTWARE ENGINEERING

S.E.R.T
reThink•reThought

PRO.M.IS
security built in

Thank you.

more: *rethought.se*
promisedu.se

contact: *Prof. Dr. Tony Gorschek*
tony.gorschek@bth.se

BTH SERL Sweden
LEADING SOFTWARE ENGINEERING

