

AI, GLAD & Organizational Design

1

1



2

2

highlighting major AI change
for devs & orgs since 2020,
not my “new fad”

AI undergrad & grad focus
since late 1970s

paid full-time AI developer &
teacher starting 1986

3

3

2020 LeSS Conference Keynote: AI Impact

Premium Search

UX Collective TOP STORIES ABOUT PUBLISH A STORY RECEIVE OUR NEWSLETTER

Let's talk about that GPT-3 AI tweet that shook designers to the core

“We're going to get automated — it's just a matter of time.”

1:19 / 28:16 • Automated Away

Learning Adaptiveness - Craig Larman

Large-Scale Scrum (LeSS) Subscribe 41 Share Download All From Large-Scale Scrum (LeSS)

4

4

My Thesis...

a 2nd-generation AI that automatically
learned from a 1st-generation AI ...

Learning From Knowledge Systems

by
Craig Larman

5

5

Demo

6

demo for tasks done
traditionally by
product mgrs,
analysts, UI/UX
designers, testers,
coders, DB designers

7

7

GLAD

8

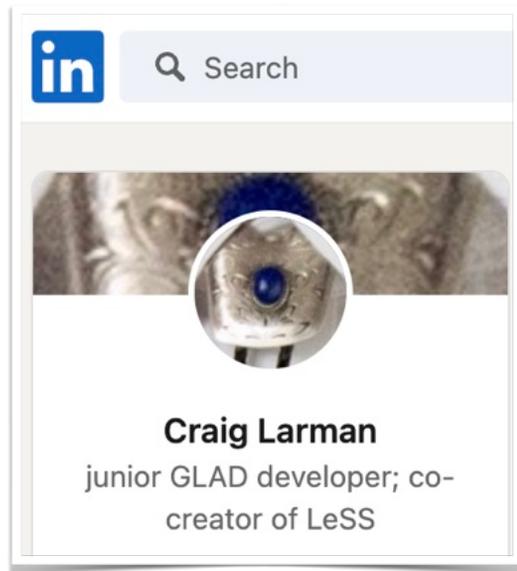
for development, we need a **short new term** for “AI tools that 1) generate, & 2) talk/teach” ...

9

GLAD

Generative-ai & LLM-Assisted Development

10



11

Social Impacts

12

for example...

Google DeepMind

Research

Accelerating fusion science through learned plasma control

February 16, 2022

MIT-designed project achieves major advance toward fusion energy

DAVID L. CHANDLER · JANUARY 12, 2022 · MIT NEWS

13

13

Org Design Impacts

14

Als are *finally* doing
“knowledge worker”
routine expertise

15

Automation Supplanting Humans

1. routine math (1940s)
2. routine movement (1940s)
3. routine info workflows (1950s)
4. **(next) routine expertise (2020s)**
>with powerful **task-specific** Als

16

Routine Single-Specialist Expertise

- > product management
- > business requirements/analysis
- > UI/UX designer
- > front end developer
- > DB designer
- > “payment service” developer
- > tester

17

17

novelty of knowledge work



routine
expertise

learning
expertise

*has/will be
automated
by AIs*

*human value-
add (for now)*

18

Routine Single-Specialist Expertise

- > product management
- > business requirements/analysis
- > UI/UX designer
- > front end developer
- > DB designer
- > “payment service” developer
- > tester

routine
expertise

*has/will be
automated
by AIs*

19

19

“AIs won’t replace
our narrow
single-specialist
routine jobs”

(says the wishful thinker)

20

20

“I will simply work more
efficiently in my single-
speciality, with AI.”

(says the wishful thinker)

21

21

really? ...

22

22

story: Lisbon coach &
automated testing

23

23

18 minutes!

**Web Summit Rio 2023: Building
an app in 18 minutes with
GitHub Copilot X**

GitHub CEO Thomas Dohmke demonstrated the power of GitHub Copilot X live on stage.

24

24

CUSTOM WEBSITES GENERATED BY A.I.

In 30 Seconds

Get a custom AI-built digital presence with Chat GPT and gain access to our

25

25

and these are stories
of baby beta-products

imagine in “3” years

26

26



BIG Idea

AI tools will allow one to
produce **10X-1000X**
faster, not just “20%”

(with much less effort,
errors, & prior expertise)

27

27

imagine a single-specialist UI/
DB designer that can now do

300 person-days of output

...in 3 or 30 person-days

28

28

300 person-days of output

->

3 or 30 person-days

what will the “UI/DB designer” do for the remaining “297” days?

29

29

what will the “UI/DB designer” do for the remaining “297” days?

perhaps they will create LOTS MORE UIs & DBs?

NO, because there is no customer demand

30

30

what will the “UI/DB designer” do for the remaining “297” days?

perhaps fire all but 1 designer, who “does all the design”?

1. **UNLIKELY**, because they still have “200” days free
2. **UNLIKELY**, because other developers can use these tools to quickly learn & create designs
3. **UNLIKELY**, because many companies will strive to avoid massive layoffs
4. **UNLIKELY**, because it would create unnecessary delays and weak feedback loops

31

31

novelty of knowledge work



routine expertise

has/will be automated by AIs

learning expertise

human value-add (for now)

32

what will the “UI/DB designer” do for the remaining “297” days?

perhaps fire all but 1 designer, who “does all the design”?

1. UNLIKELY, because they still have “200” days free

2. UNLIKELY, because other developers can use these tools to quickly learn & create designs

3. UNLIKELY, because many companies will strive to avoid massive layouts

4. UNLIKELY, because it would create unnecessary delays and weak feedback loops

33

33



BIG Idea

GLAD makes it much easier to **learn** new skills & tools

34

34

if a “UI/DB Designer” can much more easily/quickly learn & create...

automated testing
front-end dev
etc

then...

35

35



BIG Idea

1. primary
2. secondary
3. tertiary
skills learning

36

36

if “DB Designer” does DB tasks for “10” days per year, and learns/does UI design, programming, testing, etc for “290” days...

why do they have the single-specialist job title “DB Designer”?

why is the career path “DBD-1, DBD-2, ...”?

37

37



BIG Idea

structural change:
formally in HR titles:
broad job title, with career
path **multi-learning**

e.g. **Product Developer**

38

38

Then, Not a Team of *Single-Specialists*



39

39

A Team of ML PDs, with AI assistants

multilearning product developer
multilearning product developer

40

40

this is not a new idea...

41

41

Harvard Business Review

LEADING TEAMS

The New New Product Development Game

by Hirotaka Takeuchi and Ikujiro Nonaka
FROM THE JANUARY 1986 ISSUE

Moving the scrum downfield

From interviews with organization members from the CEO to young engineers, we learned that leading companies show six characteristics in managing their new product development processes:

- 1 Built-in instability
- 2 Self-organizing project teams
- 3 Overlapping development phases
- 4 "Multilearning"
- 5 Subtle control
- 6 Organizational transfer of learning

42

42

"This won't change manager roles"

(says the wishful thinker)

43

43

if there is no "DB Design" group (since no "DB Designers"), is there a...

DB Design manager?

44

44

“it’s 2023. here’s our official job categories in our modern high tech company with leading-edge HR & leadership:



business analyst
ui designer
front-end dev
db designer
...”



45

novelty of knowledge work

routine expertise

learning expertise

has/will be automated by AIs

human value-add (for now)

46

MAJOR AI Paper: Towards *Learning/Adapting AIs*

On the Measure of Intelligence

François Chollet *
Google, Inc.
fchollet@google.com
November 5, 2019

“The intelligence of a system is a measure of its skill-acquisition [learning] efficiency”

47

increasingly, job skill will be
skill in multi-learning over
skill in single-doing

48

Close

49



welcome to
connect with me!

50

structured Q&A

51

Dead Demo

52

bc teach me key points in the Medicare Promoting Interoperability Program that a programmer should know

The Medicare Promoting Interoperability Program, previously known as the Medicare EHR (Electronic Health Record) Incentive Program, is part of a broader initiative by the U.S. government to encourage healthcare providers to implement and meaningfully use EHR systems. This program provides financial incentives to eligible healthcare providers who meet certain criteria, with the ultimate goal of improving patient care through better access to and use of health data.

As a programmer, here are some of the key points you should be aware of:

1. **Understanding of EHR Systems:** EHR systems are used to store, retrieve, and manage electronic health records. They often include a range of features, from scheduling and billing to prescription management and laboratory tests. Having a solid understanding of these systems can provide the foundation for meeting the requirements of the Promoting Interoperability Program.
2. **Standards and Interoperability:** A significant part of the program emphasizes the importance of standards and interoperability. Inter

53

bc who are the top five competitors to stripe, and what are their strengths and weaknesses?

As of my knowledge cutoff in September 2021, some of the top competitors to Stripe include PayPal, Square, Braintree, Adyen, and Authorize.net. Here's a brief rundown of their strengths and weaknesses:

1. PayPal

- **Strengths:** PayPal is one of the most recognized payment gateways worldwide with a massive user base. It offers a wide array of services, including peer-to-peer payments, business payments, and mobile payments. It's quite popular with consumers, which can be a benefit for businesses looking to instill trust.
- **Weaknesses:** PayPal's fees are higher than some competitors for businesses. The user interface is not as straightforward as some other platforms, and some users complain about customer service.

2. Square

- **Strengths:** Square provides a complete set of tools for businesses, including POS systems, payment processing, and other business software. It's particularly strong in the area of mobile payments and is popular with small businesses due to its simple fee structure.
- **Weaknesses:** Square

54

bc teach me key requirements that a programmer should know when writing code for credit card processing

Writing code for credit card processing can be quite complex due to the high level of security and compliance required. Here are some key requirements that you should keep in mind:

1. **PCI Compliance:** The Payment Card Industry Data Security Standard (PCI DSS) is a set of security standards designed to ensure that all companies that accept, process, store or transmit credit card information maintain a secure environment. Any code written for credit card processing must comply with these standards.
2. **Secure Transmission:** Card details should always be transmitted securely, typically over HTTPS. This ensures that the data is encrypted and cannot be intercepted during transmission.
3. **Tokenization:** Instead of storing sensitive card data, use a process called tokenization. This involves replacing the card data with a unique identifier or "token" that is useless if stolen.
4. **No Storage of Sensitive Data:** It's a violation of PCI standards to store certain sensitive data, including the card verification

55

bc Create a checklist, in a table, for UI elements for a perfect sign-up process

Sure, here's a sample checklist in a table format:

| No. | UI Element | Description |
|-----|-----------------------------|--|
| 1 | Logo | Company/brand logo should be visible and properly placed. |
| 2 | Header | Clear and concise header stating the purpose of the page. |
| 3 | Description | A brief description of the process and its benefits. |
| 4 | Input Fields | Minimum necessary fields like email, username, and password. Ensure the form is not too long or complicated. |
| 5 | Placeholder Text | Sufficient placeholder text in input fields to guide users. |
| 6 | Validation | Real-time validation to check input format and show error/success messages. |
| 7 | Password Strength Indicator | To help the user create a secure |

56

DC use these elements to create a webpage in html and javascript

Sure, here is a simple example of a sign-up page using HTML, CSS, and JavaScript:

```
html
<!DOCTYPE html>
<html>
<head>
<style>
body {
  font-family: Arial, sans-serif;
}

.container {
  width: 300px;
  padding: 16px;
  background-color: white;
  margin: 0 auto;
  margin-top: 100px;
  border: 1px solid black;
  border-radius: 4px;
}

input[type=text], input[type=password] {
```

DC write cucumber tests for a sign-up process

Cucumber tests, also known as Behavior-Driven Development (BDD) tests, are usually written in Gherkin language. They focus on the behavior of a system from the end-user's perspective.

Let's write some Cucumber tests for a simple sign-up process.

```
gherkin
Feature: User Signup

Scenario: Successful Signup
  Given I am on the sign-up page
  When I enter valid details
  And I click the sign-up button
  Then I should be redirected to the home page
  And I should see a success notification

Scenario: Signup with existing email
  Given I am
```

DC write a wire-frame for a sign-up process

Creating a wireframe is a visual process which is often done using design software or online tools. It's not really possible to create an exact wireframe in a text-based interface like this one. However, I can describe a simple wireframe for a sign-up process:

- 1. Header (Top of Page)**
 - The website logo positioned on the left side.
 - Other navigation links on the right side.
- 2. Main Content Area (Center of Page)**
 - A large heading "Sign Up" centered on the top of the page.
 - Below the heading, three input fields aligned vertically: "Name", "Email", and "Password". Each field has a corresponding label.
 - Under the input fields, a checkbox with a label "I agree to the terms and conditions".
 - A "Sign Up" button below the checkbox.
 - A text link "Already have an account? Log in" below the "Sign Up" button.
- 3. Footer (Bottom of Page)**
 - Copyright information and additional links.

In a real-life scenario

DC write cucumber tests for credit card processing

Cucumber is a testing tool that supports Behavior Driven Development (BDD). It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called Gherkin.

In the context of credit card processing, let's assume the following user story:

```
css
As a user,
I want to be able to pay with my credit card,
So that I can easily purchase items online.
```

We can transform this into the following Cucumber scenarios:

```
gherkin
Feature: Credit Card Processing

Scenario: Successful credit card payment
  Given a user with a valid credit card
  When the user attempts
```


DC teach me an introduction to mongodb, in less than 200 words

MongoDB is a NoSQL database that provides high performance, high availability, and easy scalability. It works on the concept of collections and documents, where documents comprise sets of key-value pairs and are the basic unit of data in MongoDB. Collections contain documents and function as the equivalent of relational database tables. Unlike a SQL database, there are no relations between collections. MongoDB is schema-less, which means a document in the same collection doesn't need to have the same structure or fields. It supports a rich query language, secondary indexes, and ACID transactions. It's often used for storing semi-structured data and for executing large-scale processing of various forms of data.

65

65

DC teach me more about mongodb documents

MongoDB documents are the basic unit of data in MongoDB and are stored in BSON format, a binary representation of JSON-like documents. They can contain many different types of data including arrays, binary data, and sub-documents.

A MongoDB document is composed of field-and-value pairs and has the following structure:

```
json Copy code
{
  field1: value1,
  field2: value2,
  field3: value3,
  ...
  fieldN: valueN
}
```

The value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents.

Documents in MongoDB are similar

66

66

DC write python code that sets up using a mongodb database, and then inserts one document

To use MongoDB in Python, we need to install a driver that allows Python to interact with MongoDB. The most common driver is PyMongo. If you haven't installed it yet, you can do it with the following command:

```
bash Copy code
pip install pymongo
```

Here is a basic example of connecting to a MongoDB database and inserting a document using PyMongo:

```
python Copy code
from pymongo import MongoClient

# Create a MongoClient to the running MongoDB instance
client = MongoClient('mongodb://localhost:27017/')

# Get
```

67

67

DC write sql to define commonly used tables and fields for ecommerce

Here's an example of SQL code that defines some basic tables you'd commonly see in an eCommerce database. These tables include 'Customers', 'Products', 'Orders', and 'OrderDetails'.

```
sql Copy code
CREATE TABLE Customers (
  CustomerID INT AUTO_INCREMENT,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Address VARCHAR(255),
  City VARCHAR(50),
  PostalCode VARCHAR(20),
  Country VARCHAR(50),
  PRIMARY KEY (CustomerID)
);

CREATE TABLE Products (
  Product
```

68

68



does that schema conform to third-normal form?



Yes, the given schema generally conforms to the Third Normal Form (3NF). Here's why:

1. **First Normal Form (1NF):** In the schema, each table is organized as a relation where every column contains atomic (indivisible) values, and each row contains a unique identifier (the primary key).
2. **Second Normal Form (2NF):** Each table in the schema has its non-primary key attributes fully dependent on the primary key. For example, in the `products` table, `name`, `description`, `price`, `image_url`, and `category_id` are all dependent on the `id` (primary key).
3. **Third Normal Form (3NF):** The tables in the schema do not contain any transitive dependencies. Each non-primary attribute is non-transitively dependent on the primary key. For example, in the `orders` table, `status`, `order_date`, and `total` depend only on the `id` (primary key), not on any other non-primary attribute.

It's worth mentioning that while normalization helps reduce redundancy and improve data integrity, it may not always be the best approach for every situation, especially when performance is a concern. The balance between normalization and performance depends on the specific requirements of your application.

69

69



```
do a code review of: public static int dayOfYear(int month, int dayOfMonth, int year) {
    if (month == 2) {
        dayOfMonth += 31;
    } else if (month == 3) {
        dayOfMonth += 59;
    } else if (month == 4) {
        dayOfMonth += 90;
    } else if (month == 5) {
        dayOfMonth += 31 + 28 + 31 + 30;
    } else if (month == 6) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31;
    } else if (month == 7) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30;
    } else if (month == 8) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31;
    } else if (month == 9) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31;
    } else if (month == 10) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30;
    } else if (month == 11) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31;
    } else if (month == 12) {
        dayOfMonth += 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 31;
    }
    return dayOfMonth;
}
```

70

70



This method seems to calculate the day of the year given a date (month, day, and year). Here are a few observations and suggestions for improvement:

1. **Leap Years:** This code does not account for leap years. February has 29 days in a leap year, which would shift all subsequent months' days. You would need to check if the `year` is a leap year and adjust February's day count accordingly.
2. **Code Duplication:** This method has a significant amount of code duplication. Instead of manually summing the days of each month, you could use an array to store the number of days in each month, and sum up the days in a loop.
3. **Error Checking:** This method doesn't handle invalid inputs. What if the `month` is greater than 12 or less than 1, or if the `dayOfMonth` is outside the valid range for the given month? Adding error checks would make your code more robust.
4. **Unused Parameter:** The `year` parameter is unused. If you plan to handle leap years, you will need this parameter.

71

71

Here's a revised version of the method that addresses these issues:

```
java Copy code
public static int dayOfYear(int month, int dayOfMonth, int year) {
    if (month < 1 || month > 12 || dayOfMonth < 1 || dayOfMonth > 31) {
        throw new IllegalArgumentException("Invalid date");
    }

    int[] daysInMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Adjust for leap years
    if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)) {
        daysInMonth[2] = 29;
    }

    int dayOfYear = 0;
    for (int i = 1; i < month; i++) {
        dayOfYear += daysInMonth[i];
    }
    dayOfYear += dayOfMonth;

    return dayOfYear;
}
```

This version of the method is cleaner, handles leap years, and checks for invalid dates.

72

72