Software (design) for Data Scientists

ISEA Session 3

David Beck
University of Washington
1.31.2025

Overview of today

1. Motivate the role of intentional software design
2. Overview of a software design approach
3. Users and their stories inform design
4. Use cases describe the function of software
5. Components implement the use cases
6. Testing and testing strategies


IES
Institute of
Education Sciences

AmplifyLearn.AI

UNIVERSITY of WASHINGTON
eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

UNIVERSITY OF
OREGON

Software Design

"...specification of a software artifact, intended to accomplish goals, using a set of primitive components ..." [wikipedia]

Why design?

“I have an idea and I’m ready to code now!”

Benefits of a Software Design

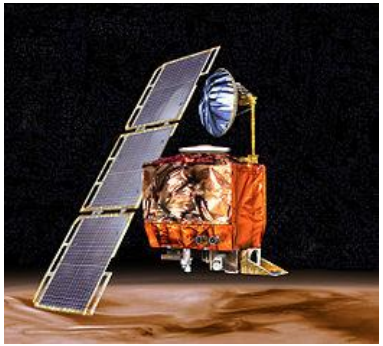
- Provides a systematic approach to a complex problem
- Find bugs before you code
- Enables many people to work in parallel
- Promotes testability
- Promotes usability

Demonstrably “true” software with features users want is going to be used.

Drawbacks of a Software Design

> Heavy lift for small tasks
> It can be impossible to know when a design is complete
> Others?

Design fails



- Mars Climate Orbiter (1998-1999)
  - $551 million in 2022 USD
  - **NASA and Lockheed Martin did not specify units in design**
    - Two separate systems interacted during injection burn
    - Foot/pounds = 1.356 Newton-meters
  - Bounced of Martian atmosphere circling the sun today

Design fails

NIH NATIONAL CANCER INSTITUTE

- caBIG 2004-2012
  - $350 million in 2022 in 2010 USD
  - CAncer Biomedical Informatics Grid
  - Unified software infrastructure for cancer data collection, analysis, management, including
    - Clinical trials from enrollment to adverse event reporting
    - Sample collection, annotation, storage and sharing of medical imaging data
    - Biospecimen management and control

IES Institute of Education Sciences

AmplifyLearn.AI

UNIVERSITY of WASHINGTON eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

UNIVERSITY OF OREGON

Design fails

NIH NATIONAL CANCER INSTITUTE

- caBIG 2004-2012
  - $350 million in 2022 in 2010 USD
  - CAncer Biomedical Informatics Grid
  - Unified software infrastructure for cancer data collection, analysis, management
  - **Who were the users? Pharma? Academic research?**
  - **Did people want the software proposed?**
  - **Competition with existing software**

IES Institute of Education Sciences

AmplifyLearn.AI

UNIVERSITY of WASHINGTON eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

UNIVERSITY OF OREGON

Design fails

NIH NATIONAL CANCER INSTITUTE

- caBIG 2004-2012
  - $350 million in 2022 in 2010 USD

- Renamed and replaced by a successor (National Cancer Informatics Program) in 2012.

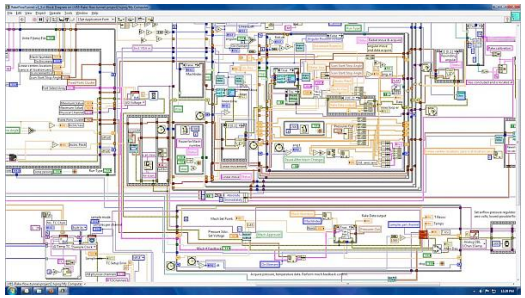
IES Institute of Education Sciences

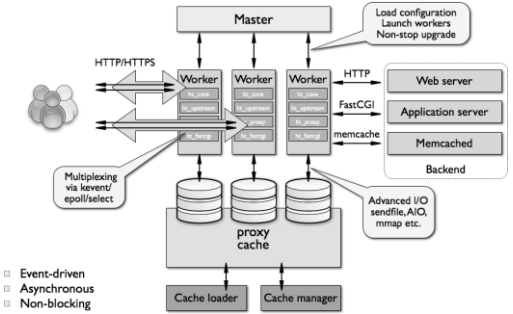
AmplifyLearn.AI

UNIVERSITY of WASHINGTON eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

UNIVERSITY OF OREGON

What makes a design understandable?





- Few components with clear roles
- Few interactions between components
- Carefully choose the features and interfaces
- Similarity with other designs
- Uses design patterns (user interfaces, parallel computing, message observers, …)

Simple where possible is better

Steps in Design[1]

Iterate. Iterate. Iterate.

1. Identify the users and their needs
2. Functional design
   - Describe what the system does (use cases)
3. Component design
   - Components are the “software artifacts” that implement the specific features of the use cases
   - Components are often hierarchical and reused

1. There are many paradigms of software design. This is one. It is focused on humans.

Running Example: Design of ATM

User stories

Design begins with your users

- Who are your users?
  - E.g. Researchers, policy makers, educators, learners, …
- What do they want to do with your software?
- How are they going to be interacting with it?
- What skill level(s) do they have and how will that impact the design?
- Information from interviews, observations, direct knowledge, best guesses

Start by writing a user story

Who is the user. What do they want to do with the tool.
What needs and desires do they want for the tool.
What is their skill level.

Start by writing a user story

Who is the user. What do they want to do with the tool.
What needs and desires do they want for the tool.
What is their skill level.

Start by writing a user story

Ram is a bank customer.

Start by writing a user story

Ram is a bank customer. Ram wants to check his balance, deposit money.

Start by writing a user story

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash.

Start by writing a user story

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM.

Start by writing a user story

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

Start by writing a user story

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

Start by writing a user story

- We may write multiple stories around similar users
  - These can reveal “importance” of certain features
- Take 2-3 minutes to sketch a user story for another “bank customer.”

  (You can’t interview anyone… Use direct knowledge: you are the user, feel free to express your frustrations with your bank!)

  Who is the user. What do they want to do with the tool. What needs and desires do they want for the tool. What is their skill level.

Start by writing a user story

Asma is a bank customer. Asma wants to check her balance and take out cash. She uses auto-deposit for her paychecks. She wants a safe and secure interface for interacting with the ATM. Asma is quite technical, but she wants to minimize her time interacting with the ATM and values a simple interface.

Start by writing a user story

Asma is a bank customer. Asma wants to check her balance and take out cash. She uses auto-deposit for her paychecks. She wants a safe and secure interface for interacting with the ATM. Asma is quite technical, but she wants to minimize her time interacting with the ATM and values a simple interface.

Start by writing a user story

> How are Ram and Asma the same?

> How are Ram and Asma different?

> What are the key takeaways from their user stories?

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

Asma is a bank customer. Asma wants to check her balance and take out cash. She uses auto-deposit for her paychecks. She wants a safe and secure interface for interacting with the ATM. Asma is quite technical, but she wants to minimize her time interacting with the ATM and values a simple interface

Start by writing a user story

- How are Ram and Asma the same?
  - **Bank customers, check balance, want safe and secure, simple user interface**
- How are Ram and Asma different?
  - **Use of cash, technical skill level**
- What are the key takeaways from their user stories?
  - **Safety, security and simplicity of the UI**

Start by writing a user story

- Other user stories?

- You may have several different kinds of “users”
- There may be a “technician”
- There may be a “systems administrator”

Start by writing a user story

Valentina is an ATM technician. She services ATMs as part of preventative maintenance, applies hardware and software updates and for performs emergency repairs. For maintenance and updates she will follow a standard protocol. For repairs, she needs access to a diagnostic interface. Valentina is highly technical and knows how to replace standardized parts.

Start by writing a user story

Valentina is an ATM technician. She services ATMs as part of preventative maintenance, applies hardware and software updates and for performs emergency repairs. For maintenance and updates she will follow a standard protocol. For repairs, she needs access to a diagnostic interface. Valentina is highly technical and knows how to replace standardized parts.

Start by writing a user story

- Other user stories?

- You may have several different kinds of “users”
- There may be a “technician”
  - Train a machine learning model to be used by the system, manage external data sources
- There may be a “systems administrator”
  - Creates user accounts, sends maintenance announcements

Start by writing a user story

- Other user stories?

- Thieves, scammers, safe crackers, ne'er-do-wells
  - How do you design “against” these types of “users?”

Use Cases

Functional Design

IES Institute of Education Sciences
AmplifyLearn.AI
UNIVERSITY of WASHINGTON eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS
UNIVERSITY OF OREGON

Running Example: Design of ATM

How to find use cases? In the user stories!

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

How to find use cases? In the user stories!

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

How to find use cases? In the user stories!

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

- Check balances
- Deposit checks

How to find use cases? In the user stories!

Ram is a bank customer. Ram wants to check his balance, deposit money. He rarely uses cash. Ram wants a safe and secure interface for interacting with the ATM. Ram's job does not involve technical skills and he values a simple user interface.

- Check balances
- Deposit checks

What do we do with ATMs?

- Check balances
- Deposit checks

Ram and Asma

- Get cash

Asma

- **These are examples of *Use Cases*.**
- **They describe the functional potential of software.**

Describing a Use Case (one way)

- What are the inputs and what are the outputs?

- **Adding two numbers**
  - **Take a minute to think about this use case**
  - What are the input(s)?
  - What are the output(s)?
  - How does the use case transform input to output?

Describing a Use Case (one way)

- What are the inputs and what are the outputs?

- **Adding two numbers**
  - **Inputs: two numbers**
  - **Outputs: one number, the sum of the two inputs**

  - Can we add more detail? What kind of numbers? Positive **and** negative?

Describing a Use Case (Check Balance)

- What are the inputs and what are the outputs?

- **Check balance**
  - **Take a minute to think about this use case**
  - What are the input(s)?
  - What are the output(s)?
  - How does the use case transform input to output?

Describing a Use Case (Check Balance)

- What are the inputs and what are the outputs?

- **Check balance**
  - **Input:** User selects an account
  - **Output:** ATM displays the current account balance
  - The account information is looked up in the account database and the current balance is retrieved.

Implied use cases are important!

- Some uses cases are implied
  - Involved without being a primary use case
  - “Easily overlooked”
- **What might be an implied use case of an ATM?**

Implied use cases are important!

- Authentication is an implied use case.

- **Do we need this use case?**
- **What users is authentication for?**

- **Take 2-3 minutes to describe the authentication use case.**

Describing a Use Case (Authentication)

- What information the user provides (inputs)
- What responses the system provides (outputs)

Authenticate User Use Case

User: Put ATM card in reader
ATM: Display 'Enter PIN'
User: Enters PIN on keyboard
ATM: [if correct] Show main menu
[if incorrect] Display 'Enter PIN'

Component Design

IES Institute of Education Sciences
AmplifyLearn.AI
UNIVERSITY of WASHINGTON eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS
UNIVERSITY OF OREGON

What is a component?

- Software (or other kinds) components “do the work”
- Components store data
- Components calculate values
- Components “interact” with each other
- Components “interact” with the user
- Components can be functions, databases, interfaces, external web sites, ..

Component design or component spec

- Breaks down use cases into the required components.

- Describe components in sufficient detail for someone else (or **ChatGPT***) to write the code.

* ChatGPT writes much of the first pass code in my research team based on our text descriptions of the components.‡

https://github.com/EvanKomp/aide/blob/main/docs/component_specification.md

Specification of a component

- Describe components with sufficient detail so that someone with modest knowledge of the project can implement the code for the component.
  - Name
  - What it does
  - Inputs (with type information)
  - Outputs (with type information)
  - How it uses other components
  - Side effects

Developing component specifications

1. Use case by use case: what are the components required for this use case?
   - A component is a distinct entity that “does the work”

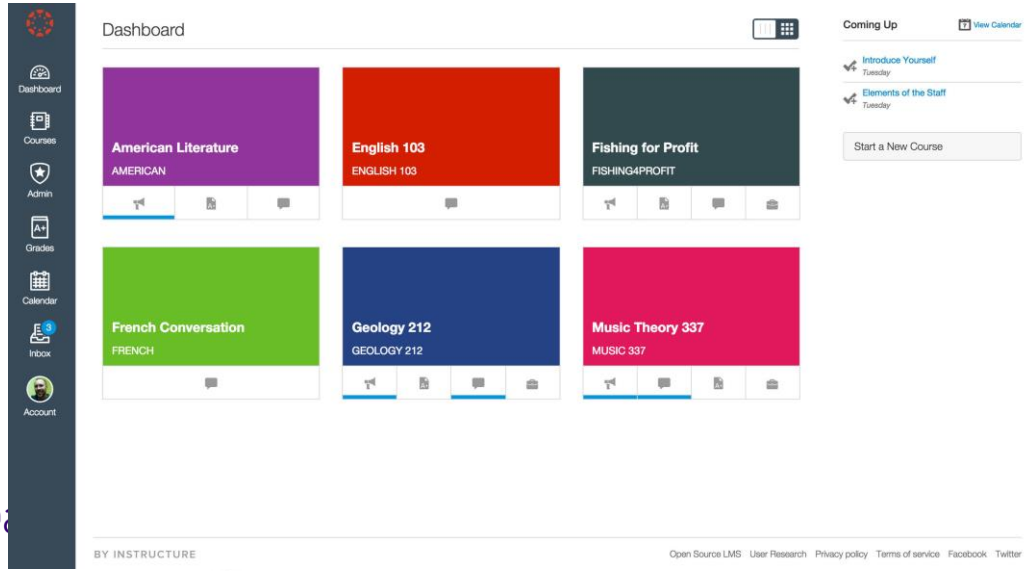
   - **What are some components on an ATM?**
     - > Display, keypad, card reader, camera
     - > Cash dispenser, envelope reader

Developing component specifications

1. Use case by use case: what are the components required for this use case?
2. Are those components used for another use case?
   - **Good, we can reuse them!**
3. Can the component be further divided in complexity for sub-components?
   - **Good, we can simplify them!**

Subcomponents can be confusing

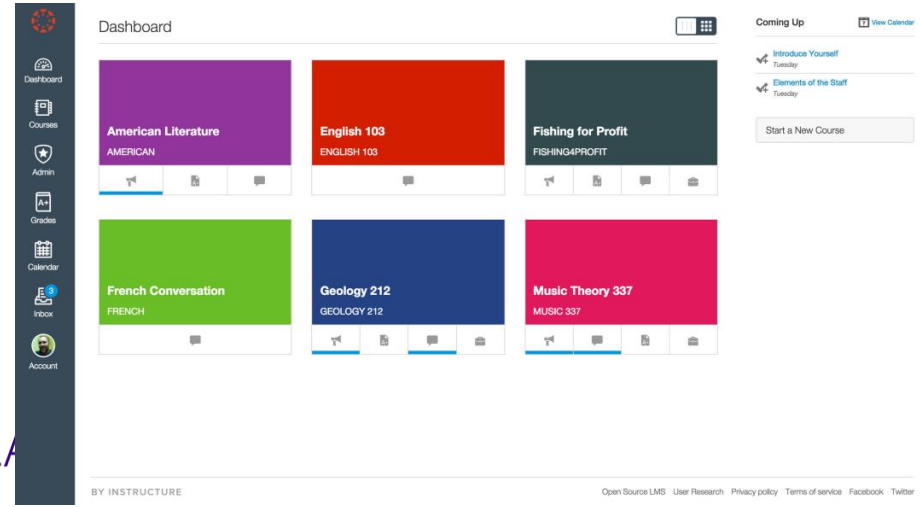
- I really don't like all your fancy vocabulary.
- **What kinds of components have subcomponents?**
  - User interfaces
    - Subcomponents are buttons, sidebars
  - Databases
    - Student table
    - Teacher table
    - Grades table



Subcomponents can be confusing

- What kinds of components have subcomponents?
  - User interfaces, databases, application programming interfaces
- **When do I need to define a subcomponent?**
  - Top down or bottom up
  - Goal: simple is better

Does subdividing the component simplify building or testing the component?



IES Institute of Education Sciences

AmplifyLearn.A

ATM components by Use Case

- Authenticate user

- Take 1 minute to think about the components in the authenticate user use case. What components can you identify?

Authenticate User Use Case

User: Put ATM card in reader
ATM: Display 'Enter PIN'
User: Enters PIN on keyboard
ATM: [if correct] Show main menu
[if incorrect] Display 'Enter PIN'

ATM components by Use Case

- **Authenticate user**
  - Database with user info including ATM card # and PIN
  - Card reader that reads ATM card
  - User interface that shows information (80x24)
  - User interface that reads user PIN
  - Authenticate control logic
    - What is this?

1

1. https://www.deviantart.com/love-rainbowflower/art/Magic-Happens-Here-lineart-292524154

ATM components by Use Case

> Check balance

> Take 1 minute to think about the components in the check balance use case. What components can you identify?

Check balance

- **Input:** User selects an account
- **Output:** ATM displays the current account balance
- The account information is looked up in the account database and the current balance is retrieved.

ATM components by Use Case

- **Check balance**
  - Database with user info including account balances
  - User interface that reads account selection
  - User interface that shows information (80x24)
  - Check balance control logic

Identify shared components

> **Authenticate user**
  - Database with user info including ATM card # and PIN
  - Card reader that reads ATM card
  - User interface that shows information (80x24)
  - User interface that reads user PIN
  - Authenticate control logic

> **Check balance**
  - Database with user info including account balances
  - User interface that reads account selection
  - User interface that shows information (80x24)
  - Check balance control logic

> **User interface (output)?**

> **User interface (input)?**

> **Database?**

> **Control logic?**

Identify shared components

- **Authenticate user**
  - Database with user info including ATM card # and PIN
  - Card reader that reads ATM card
  - User interface that shows information (80x24)
  - User interface that reads user PIN
  - Authenticate control logic

- **Check balance**
  - Database with user info including account balances
  - User interface that reads account selection
  - User interface that shows information (80x24)
  - Check balance control logic

- **User interface (output)? Yes!**
- **User interface (input)? Yes!**
- **Database? After subcomponents.**
- **Control logic? No!**

- Name
- What it does
- Inputs (with type information)
- Outputs (with type information)
- How it uses other components
- Side effects

Specify components

> Authenticate user control logic

> Take 2-3 minutes to sketch out a specification for the authenticate user control logic component.

- Name
- What it does
- Inputs (with type information)
- Outputs (with type information)
- How it uses other components
- Side effects

Specify components

Name: **Authenticate user control logic**

What it does:

- Verifies a user is in the database and that the PIN supplied by the user matches the PIN in the database

Inputs (with type information)

- *Card number,* a **string** that is the user's card number
- *PIN*, an integer

Outputs (with type information)

- Boolean: True if success, False if failure

Components used: ATM card reader supplies *Card number* input, user inputs *PIN* via keypad, verification is performed by database

Side effects: If successful, all other bank customer use cases are enabled for the User matching the *Card number.*

- Name
- What it does
- Inputs (with type information)
- Outputs (with type information)
- How it uses other components
- Side effects

Specify components

> Check balance control logic

> Take 2-3 minutes to sketch out a specification for the check balance control logic component.

- Name
- What it does
- Inputs (with type information)
- Outputs (with type information)
- How it uses other components
- Side effects

Specify components

Name: **Check balance control logic**

What it does:

- Looks up a user's account that they provided in the account database and returns the current balance associated with that account in the database.

Inputs (with type information)

- *Account number,* a **string** that is the user's account number

Outputs (with type information)

- False if account does not exist or a floating-point number equal to the account balance

Components used: User selects an *Account Number* shown on the display with keypad input and the verification and balance lookup is provided by the database

Side effects: None.

Overview of today

1. Motivate the role of intentional software design
2. Overview of a software design approach
3. Users and their stories inform design
4. Use cases describe the function of software
5. Components implement the use cases
6. Testing and testing strategies

Component specifications get you to code

- Code needs review, testing continuously

- **Has this happened to you:**
  - You wrote some code. It works! You are happy.
  - You add a neat little feature. You think it works. You are happy.
  - It doesn't work, You found out before anything bad happened.

What is software testing?

- Code that checks if other code (code under test) is working properly
- If the “code under test”
  - Runs successfully
  - Fails gracefully (as expected, when expected)
- The tests pass and the code is “accepted” as working

What is software testing?

```
def code_under_test(a, b):
    return a + b
```

What is the code under test?
What is the code under test doing?
Could we name the code under test function better?

What is software testing?

```
def add(a, b):
    return a + b
```

```
if add(0, 0) == 0:
    return True
else:
    return False
```

What is a test we can do for add?

Is this sufficient?

What is software testing?

```
def add(a, b):
    return a + b
```

```
if add(0, 1) == 1:
    return True
else:
    return False
```

Can we test a different a, b pair?

Is this sufficient?

```
if add(0, 2) == 2:
    return True
else:
    return False
```

What is software testing?

```
def add(a, b):
    return a + b
```

```
for i in range(10):
    if add(0, i) != i:
        return False
```

Can we test many a, b pairs?

Is this sufficient?

Testing is hard! Testing is fun! **Testing can drive development!**

What is continuous software testing?

```
add(a, b)
multiply(a, b)
```

} Version 1 of our simple math library

In version 2, we want to support complex numbers. How can we be sure that our changes don't break things?

"Continuous integration" or continuously integrating new code into your software **after** testing.

The tests pass and the code is "accepted" as working

Homework for next week

- Can you imagine a piece of software that is missing from your Ed-Tech workflow, LMS, or some other software that you need? If not, please use your favorite cell phone application as an example
- What are the top three user stories that would be necessary to begin to lay out a software design?
- Can you create a set of use cases for each user story?
- How do those use cases derived from different user stories overlap?
- Which use cases will create the biggest design and testing challenges downstream because of their complexity? Can you identify a complex component that should be subdivided?

Interaction diagrams