

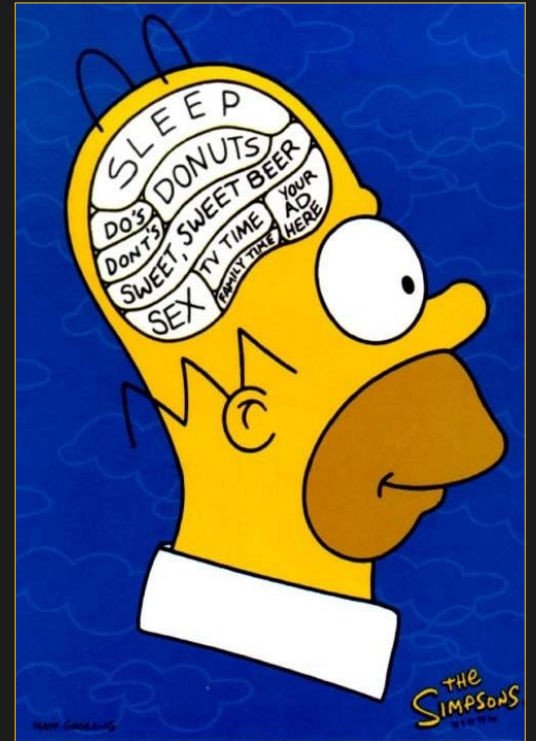
Cognitive Science based Software Engineering

What is Cognitive Science?

“Cognitive science is the study of the human mind and brain, focusing on how the mind represents and manipulates knowledge and how mental representations and processes are realized in the brain. Conceiving of the mind as an abstract computing device instantiated in the brain, cognitive scientists endeavor to understand the mental computations underlying cognitive functioning and how these computations are implemented by neural tissue.”¹

What is Cognitive Science?

Cognitive science explores how
brains process information



Memory

- Long Term Memory (LTM)
- Short Term Memory (STM)
- Working Memory (WM)

Short Term Memory

Capacity:

- 7 (± 2) items ¹
- 4 items (± 1) ²
- number depends on their category

Whatever the number is, it is limited

¹ The magical number seven, plus or minus two, by G. A. Miller

² The magical number 4 in short term memory, by N. Cowan

Short Term Memory

Duration

- 15-30 seconds ¹

Association and rehearsal make it become Long Term Memory

¹ The control process of short-term memory by R.C. Atkinson and R.M. Shiffrin

Short Term Memory

- sequence of numbers

95865536

Short Term Memory

- sequence of numbers

95865536

- works better in patterns

9586 5536

Short Term Memory

- sequence of numbers

95865536

- works better in patterns

9586 5536

Memorize it!

Working Memory

- very close to STM
- it's STM applied to processing

Working Memory

- very close to STM
- it's STM applied to processing

$16 \times 9 = ?$

Working Memory

- very close to STM
- it's STM applied to processing

$$16 \times 9 = 144$$

$$23 \times 68 = ?$$

Working Memory

- very close to STM
- it's STM applied to processing

$$16 \times 9 = 144$$

$$23 \times 68 = 1564$$

Do you remember the number?

Do you remember the number?

95865536

Do you remember the number?

95865536

9.58"

100 meters WR

65,536

Max val of a word: 2^{16}

Long Term Memory

Procedural / Implicit



Declarative / Explicit

Episodic



Semantic



Long Term Memory

```
graph TD; LTM[Long Term Memory] --> PI[Procedural / Implicit]; LTM --> DE[Declarative / Explicit]; DE --> E[Episodic]; DE --> S[Semantic];
```

Procedural / Implicit

- Touch Typing
- Vim users: ESC : q !
- Closing a tab on a browser: Ctrl +w

Declarative / Explicit

Episodic

- Coding interview
- First day on a new job

Semantic

- ArrayList class
- Binary Search
- Singleton

Chunking

Study on chess showed that experts think in patterns ¹

Chunking allows to overcome the limits of STM / WM using LTM

¹ Perception in chess, by W. Chase and H. Simon

Cognitive load ¹

Cognitive load measures the amount of mental processing required for performing a task

The higher the cognitive load, the higher the fatigue for your brain

Cognitive load types

- intrinsic
- extraneous
- germane

Intrinsic Cognitive Load

```
int abs(int a) {  
    if (a < 0) {  
        return -a;  
    }  
  
    return a;  
}
```

Intrinsic Cognitive Load

```
int abs(int a) {  
  
    if (a < 0) {  
        return -a;  
    }  
  
    return a;  
}
```

```
int gcd(int a, int b) {  
  
    if (b == 0) {  
        return a;  
    }  
  
    return gcd(b, a % b);  
}
```

Intrinsic Cognitive Load

```
int abs(int a) {  
  
    if (a < 0) {  
        return -a;  
    }  
  
    return a;  
}
```

Low

```
int gcd(int a, int b) {  
  
    if (b == 0) {  
        return a;  
    }  
  
    return gcd(b, a % b);  
}
```

High

Extraneous Cognitive Load

```
int unknown(int[] arr, int x) {int
    l=0;int r=arr.length-1;while
        (l≤r){int mid=l+(r-l)/2;
            if(arr[mid]==x){return
                mid;}if(arr[mid]>x)
                    {r=mid-1;}else{
                        l= mid+1;}
                return
                    -1 ;
            }
```

Extraneous Cognitive Load

```
int unknown(int[] arr, int x) {int
    l=0;int r=arr.length-1;while
        (l≤r){int mid=l+(r-l)/2;
            if(arr[mid]==x){return
                mid;}if(arr[mid]>x)
                    {r=mid-1;}else{
                        l= mid+1;}
                return
                    -1 ;
            }
```

```
int unknown(int[] arr, int x)  {
    int l = 0;
    int r = arr.length - 1;

    while (l ≤ r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x) {
            return mid;
        }

        if (arr[mid] > x) {
            r = mid - 1;
        } else {
            l = mid + 1;
        }

        return -1;
    }
```

Extraneous Cognitive Load

```
int unknown(int[] arr, int x) {int
    l=0;int r=arr.length-1;while
        (l≤r){int mid=l+(r-l)/2;
            if(arr[mid]==x){return
                mid;}if(arr[mid]>x)
                    {r=mid-1;}else{
                        l= mid+1;}
                return
                    -1 ;
            }
```

High

```
int unknown(int[] arr, int x)  {
    int l = 0;
    int r = arr.length - 1;

    while (l ≤ r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x) {
            return mid;
        }

        if (arr[mid] > x) {
            r = mid - 1;
        } else {
            l = mid + 1;
        }

        return -1;
    }
```

Low

Germane Cognitive Load

The effort we make transitioning from STM to LTM

What's the impact on how we deal with code?

Reading Code

We spend 58% of our time reading code rather than writing it ¹

Ok, let's see how good you're at.

¹ Measuring Program Comprehension by Xin Xia et al., 2017

What does this program computes?

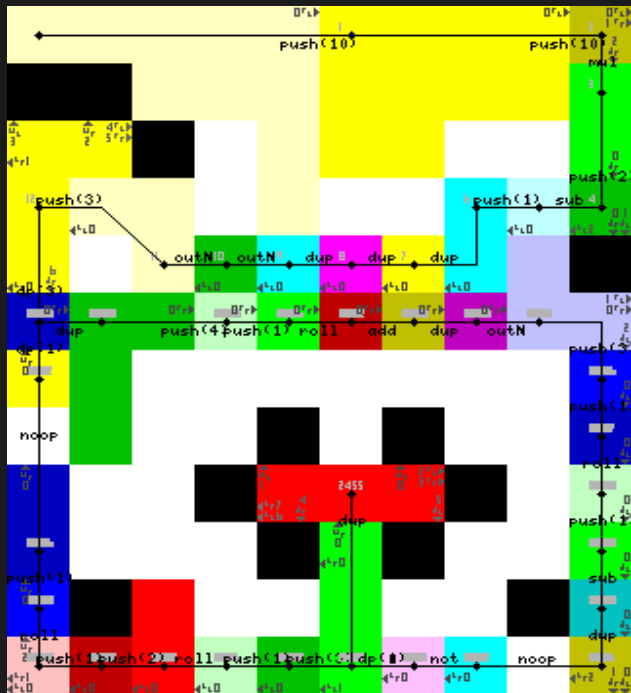


What does this program computes?



It computes a Fibonacci number
in Piet programming language

What does this program computes?



Hue Change	Lightness Change		
	No change	1 darker	2 darker
No change	N/A	Push	Pop
1 step	Add	Subtract	Multiply
2 steps	Divide	Modulo	Not
3 steps	Greater	Pointer	Switch
4 steps	Duplicate	Roll	Input num
5 steps	Input char	Output num	Output char

Light red (#FFC0C0)	Light yellow (#FFFFC0)	Light green (#C0FFC0)	Light cyan (#C0FFFF)	Light blue (#C0C0FF)	Light magenta (#FFC0FF)
Red (#FF0000)	Yellow (#FFFF00)	Green (#00FF00)	Cyan (#00FFFF)	Blue (#0000FF)	Magenta (#FF00FF)
Dark red (#C00000)	Dark yellow (#C0C000)	Dark green (#00C000)	Dark cyan (#00C0C0)	Dark blue (#0000C0)	Dark magenta (#C000C0)

Reading Code

This time for real

Reading Code

```
public int lengthOfLongestSubstring(String s) {  
    Set<Character> set = new HashSet<>();  
    int n = s.length();  
    int maxLength = 0;  
    int left = 0;  
  
    for (int right = 0; right < n; right++) {  
        while (set.contains(s.charAt(right))) {  
            set.remove(s.charAt(left));  
            left++;  
        }  
        set.add(s.charAt(right));  
        maxLength = Math.max(maxLength, right - left + 1);  
    }  
  
    return maxLength;  
}
```

Roles of a variable ¹

Fixed value	its value is not changed in run-time (after initialization)
Stepper	goes through a succession of values in some systematic way
Most-recent holder	its value is not changed in run-time after initialization
Most-wanted holder	is the "best" or otherwise the most-wanted value out of the values gone through so far
Gatherer	accumulates all the values gone through so far
Follower	gets the old value of another known variable as its new value
One-way flag	is a Boolean variable which once changed cannot get its original value anymore
Temporary	its value is always needed only for a very short period
Organizer	is used for reorganizing its elements after initialization
Container	is a data structure where elements can be added and removed
Walker	traverses a data structure

¹ Roles of Variables in Experts' Programming Knowledge, by J. Sajaniemi and R. Navarro Prieto

Reading Code

Container

```
public int lengthOfLongestSubstring(String s) {  
    Set<Character> set = new HashSet<>();
```

Fixed value

```
    int n = s.length();
```

```
    int maxLength = 0;
```

Most wanted holder

Stepper

```
    int left = 0;
```

Stepper

```
    for (int right = 0; right < n; right++) {
```

```
        while (set.contains(s.charAt(right))) {
```

```
            set.remove(s.charAt(left));
```

```
            left++;
```

```
        }
```

```
        set.add(s.charAt(right));
```

```
        maxLength = Math.max(maxLength, right - left + 1);
```

```
    }
```

```
    return maxLength;
```

```
}
```

Reading Code

We create a mental model of the algorithm and reason about it:

- edge cases**
- robustness**
- memory consumption**

Reading Code

```
public class MovingAverageCalculator {  
    private int count = 0;  
    private double average = 0.0;  
    private double sum = 0.0;  
  
    public void add(double number) {  
        sum += number;  
        count++;  
        average = sum / count;  
    }  
  
    public double getAverage() {  
        return average;  
    }  
}
```

Reading Code

```
public class MovingAverageCalculator {  
    private int count = 0;  
    private double average = 0.0;  
    private double sum = 0.0;  
  
    public void add(double number) {  
        sum += number;  
        count++;  
        average = sum / count;  
    }  
  
    public double getAverage() {  
        return average;  
    }  
}
```

Now read this code knowing that an instance of this class is accessed by multiple threads

Mental Models

- Algorithms
- System architectures
- Domains

Mental Models

Tennis Rules¹

Scoring:

- Points: 0, 15, 30, 40, Game
- Win a game: Score 4 points and be 2 points ahead
- Win a set: Win at least 6 games, leading by 2
- Win the match: Win 3 out of 5 sets

Given an array of scored points [p1, p2, p1, p1, p2, ...], find the winner

Mental Models

**Diverse teams can help in approaching
problems from different angles**

Reading Code

Seniority leverages chunking

Writing code

A lot of good practices lower cognitive load

- split long methods/functions
- single responsibility principle
- early optimization

Writing code

Side effects

```
public void setStatus(Status status) {  
    this.status = status;  
    if (status == Status.SHIPPED) {  
        sendEmailToCustomer();  
    }  
}
```

You use chunking, but it's wrong

Naming Conventions

Name	Description	Example of flawed identifier(s)
Capitalisation Anomaly	Identifiers should be appropriately capitalised.	HTML <code>EditorKit</code> .pagecounter
Consecutive Underscores	Consecutive underscores should not be used in identifier names.	foo__bar
Dictionary Words	Identifier names should be composed of words in dictionary and abbreviations, and acronyms, that are more commonly used than the unabbreviated form.	strlen
Excessive Words	Identifier names should be composed of no more than four words or abbreviations.	floatToRawIntBits()
Enumeration Identifier Declaration Order	Unless there are compelling and obvious reasons otherwise, enumeration constants should be declared in alphabetical order.	enum Card {ACE, EIGHT, FIVE, FOUR, JACK, KING ...}
External Underscores	Identifiers should not have either leading or trailing underscores.	_foo_
Identifier Encoding	Type information should not be encoded in identifier names using Hungarian notation or similar	iCount;
Long Identifier Name	Long identifier names should be avoided where possible.	getPolicyQualifiersRejectedNaming
Convention Anomaly	Identifiers should not consist of non-standard mixes of upper and lower case characters.	FOO_bar
Number of Words	Identifiers should be composed of between two and four words.	ArrayOutOfBoundsException.name
Numeric Identifier Name	Identifiers should not be composed entirely of numeric words or numbers.	FORTY_TWO
Short Identifier Name	Identifiers should not consist of fewer than eight characters, with the exception of: c, d, e, g, i, in, inOut, j, k, m, n, o, out, t, x, y, z	name

Comments

Comments are another tool for lowering the cognitive load

9 types of comments ¹

- **Function comments**
- **Design comments**
- **Why comments**
- **Teacher comments**
- **Checklist comments**
- **Guide comments**
- **Trivial comments**
- **Debt comments**
- **Backup comments**

¹ Writing system software: code comments, by S. Sanfilippo


```
if (idle > server.repl_backlog_time_limit) {  
    /* When we free the backlog, we always use a new  
    * replication ID and clear the ID2. This is needed  
    * because when there is no backlog, the master_repl_offset  
    * is not updated, but we would still retain our replication  
    * ID, leading to the following problem:  
    *  
    * 1. We are a master instance.  
    * 2. Our replica is promoted to master. It's repl-id-2 will  
    *    be the same as our repl-id.  
    * 3. We, yet as master, receive some updates, that will not  
    *    increment the master_repl_offset.  
    * 4. Later we are turned into a replica, connect to the new  
    *    master that will accept our PSYNC request by second  
    *    replication ID, but there will be data inconsistency  
    *    because we received writes. */  
    changeReplicationId();  
    clearReplicationId2();  
    freeReplicationBacklog();  
    serverLog(LL_NOTICE,  
        "Replication backlog freed after %d seconds "  
        "without connected replicas.",  
        (int) server.repl_backlog_time_limit);  
}
```

```
// check if the restaurant actually exists
if (restaurant == null) {
    throw new RestaurantNotFoundException(booking.getRestaurantId());
}

// check if the number of diners in the booking is more than the number of seats in the restaurant
if (restaurant.capacity() < booking.getNumberOfDiners()) {
    throw new NoAvailableCapacityException("Number of diners exceeds available restaurant capacity");
}

// check the restaurant is open on the day of the booking
if (!restaurant.openingDays().contains(booking.getDate().getDayOfWeek())) {
    throw new RestaurantClosedException("Restaurant is not open on: " + booking.getDate());
}

// find all the bookings for that day and check that with all the booked diners the restaurant still has
// space for the new booking diners
List allByRestaurantIdAndDate = repository.findAllByRestaurantIdAndDate(booking.getRestaurantId(),
    booking.getDate());
int totalDinersOnThisDay = allByRestaurantIdAndDate.stream().mapToInt(Booking::getNumberOfDiners).sum();
if (totalDinersOnThisDay + booking.getNumberOfDiners() > restaurant.capacity()) {
    throw new NoAvailableCapacityException("Restaurant all booked up!");
}

// if we got this far, the booking is valid and we can save it
return repository.save(booking);
```

What can happen when we write code?

The Flow ¹

Hard to reach, easy to lose

Pure focus, time flies, happiness and satisfaction

Takes 15 minutes to reach ²

¹ Flow – The psychology of optimal experience by M. Csikszentmihalyi, 1990

² Peopleware: productive projects and teams by T. DeMarco & T. Lister, 1999

The Flow

How to keep being in it

Avoid distractions:

- signal you're concentrated (status on slack/teams/etc)
- shut down notification (yes, on your mobile too)
- in office: hoodies / hat / earphones
- use fullscreen / no distraction mode

The Flow

How to keep being in it

Organize your time:

- **arrange with the team for reserved coding time**
- **early bird or night owl?**
- **setup no meeting days**
- **maintain focus with pomodoro technique?**

The Flow

How to keep being in it

Reduce tedious tasks:

- Use IDE / GenAI for writing trivial code

Testing

Immediate feedback

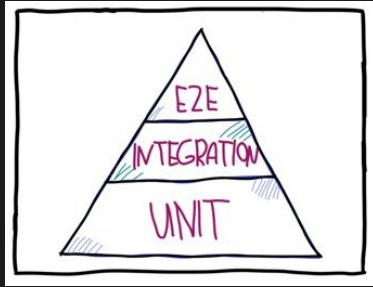
Ideally, when writing code you have:

- **unit testing below 10 secs (modularize)**
- **integration testing below 30 seconds**

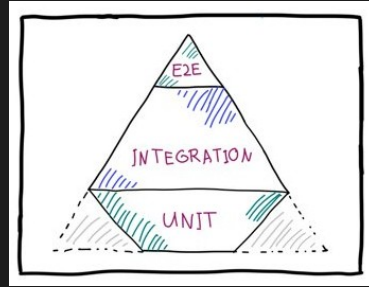
At a later stage you can have:

- **E2E testing on CI/CD pipelines**
- **Smoke tests: after deploy**

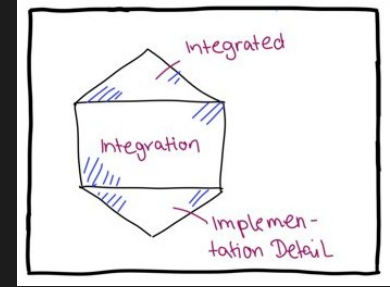
Testing Strategies



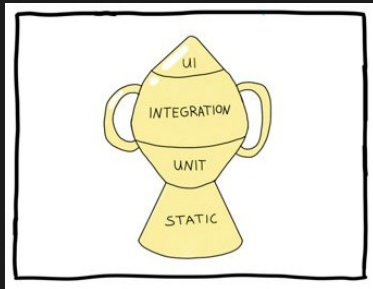
Pyramid



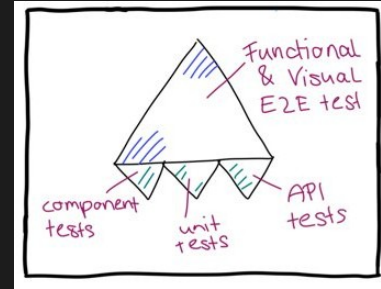
Diamond



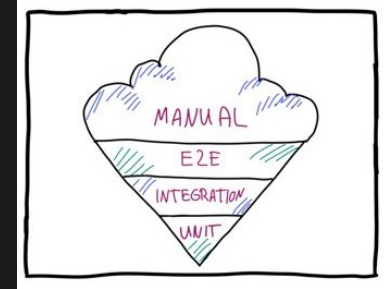
Honeycomb



Trophy



Crab



Ice Cone

We are all different

What works for me, may not work for you.

Experiment with your team (and do it whenever a new hire joins)!

But, most of all, the brain needs



By the way, do you remember the number?

By the way, do you remember the number?

95865536

Questions?

Download this presentation here:

