



Expertise in Software Development

Towards an Interdisciplinary Theory

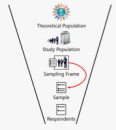
Dr. Sebastian Baltes

 @s_baltes

 [empirical-software.engineering](https://github.com/empirical-software-engineering)



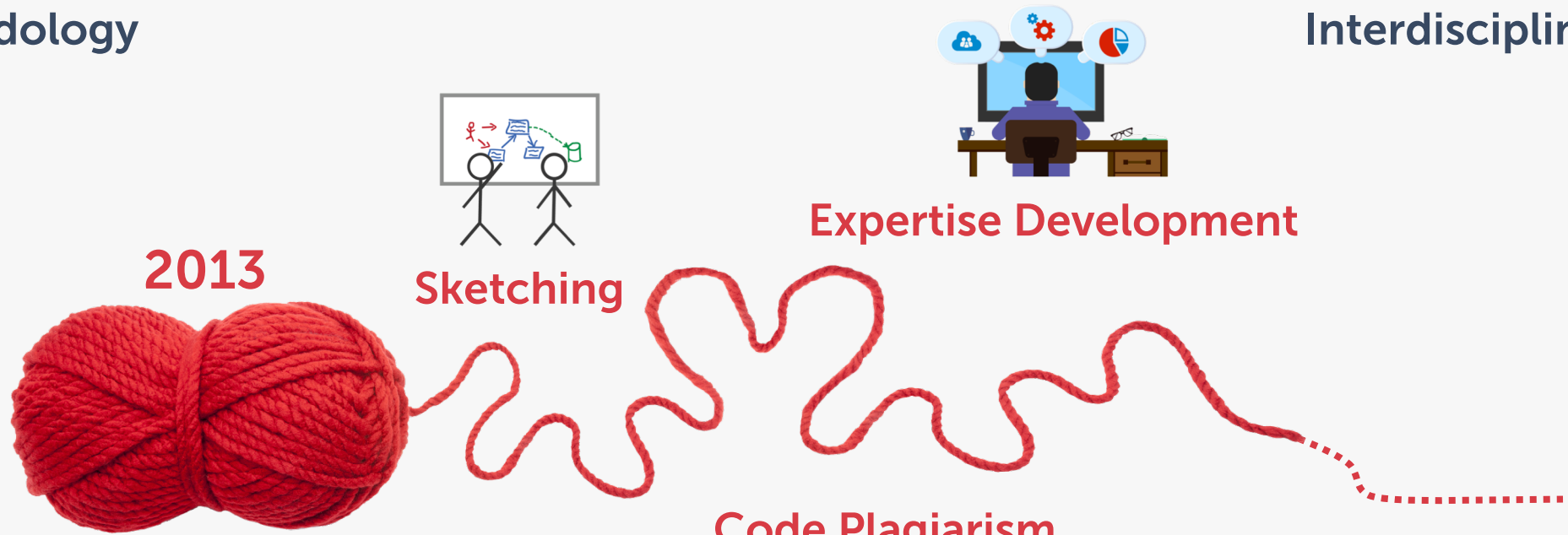
THE UNIVERSITY
of ADELAIDE



Issues in Sampling Software Developers Methodology



Constructing Urban Tourism Space Digitally Interdisciplinary Research



Code Plagiarism



Regular Expressions
RegViz

Continuous Integration



Interaction



My Background





THE UNIVERSITY
of ADELAIDE

My Background

Adelaide is ranked among
ten **most liveable cities**
in the world

- Ranked among the **top 1%** universities worldwide
- Member of the **Group of Eight**, Australia's coalition of world-leading research-intensive universities
- School of **Computer Science** is ranked among the **top 100** worldwide



My Background





Towards a Theory of Software Development Expertise

Sebastian Baltes
University of Trier
Trier, Germany
research@sbaltes.com



Stephan Diehl
University of Trier
Trier, Germany
diehl@uni-trier.de

ABSTRACT

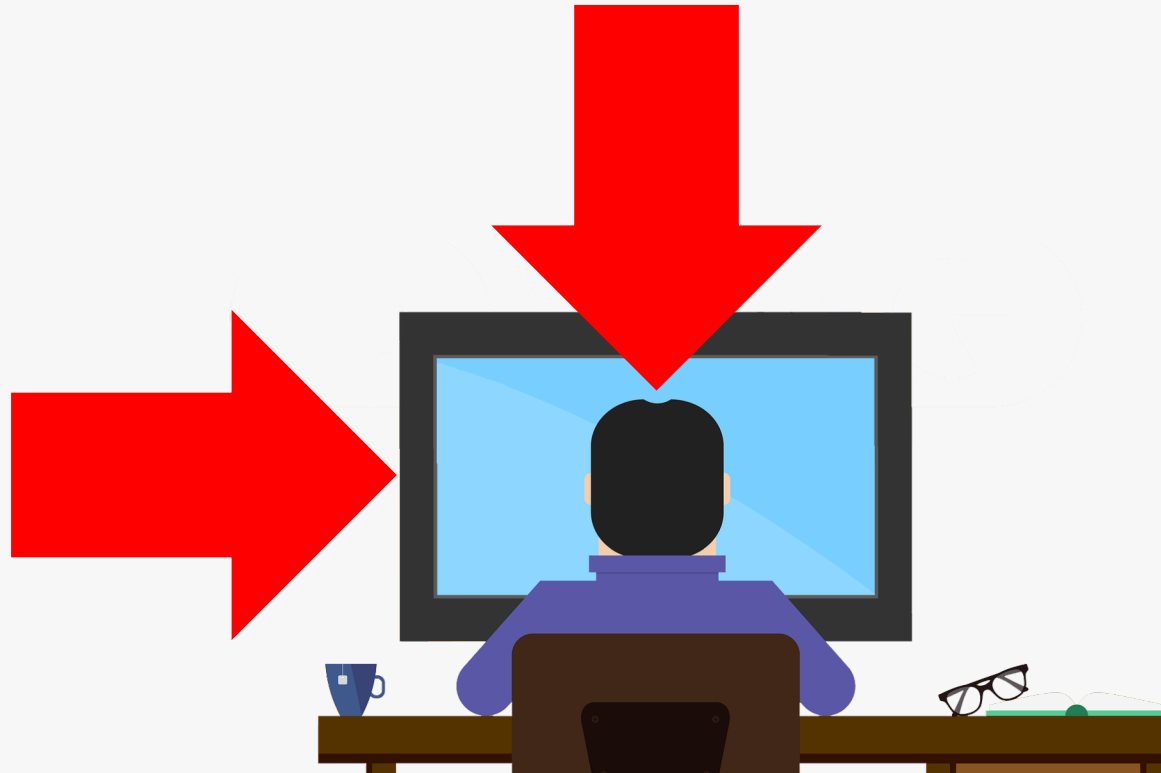
Software development includes diverse tasks such as implementing new features, analyzing requirements, and fixing bugs. Being an expert in those tasks requires a certain set of skills, knowledge, and experience. Several studies investigated individual aspects of software development expertise, but what is missing is a comprehensive theory. We present a first conceptual theory of software development expertise that is grounded in data from a mixed-methods survey with 335 software developers and in literature on expertise and expert performance. Our theory currently focuses on programming, but already provides valuable insights for researchers, developers, and employers. The theory describes important properties of software development expertise and which factors foster or hinder its formation, including how developers' performance may decline over time. Moreover, our quantitative results show that developers' expertise self-assessments are context-dependent and that experience is not necessarily related to expertise.

expert performance [78]. Bergersen et al. proposed an instrument to measure programming skill [9], but their approach may suffer from learning effects because it is based on a fixed set of programming tasks. Furthermore, aside from programming, software development involves many other tasks such as requirements engineering, testing, and debugging [62, 96, 100], in which a software development expert is expected to be good at.

In the past, researchers investigated certain aspects of software development expertise (SDExp) such as the influence of programming experience [95], desired attributes of software engineers [63], or the time it takes for developers to become “fluent” in software projects [117]. However, there is currently no theory combining those individual aspects. Such a theory could help structuring existing knowledge about SDExp in a concise and precise way and hence facilitate its communication [44]. Despite many arguments in favor of developing and using theories [46, 56, 85, 109], theory-driven research is not very common in software engineering [97].

<https://empirical-software.engineering/projects/expertise/>

Disclaimer



Software Development Expertise?

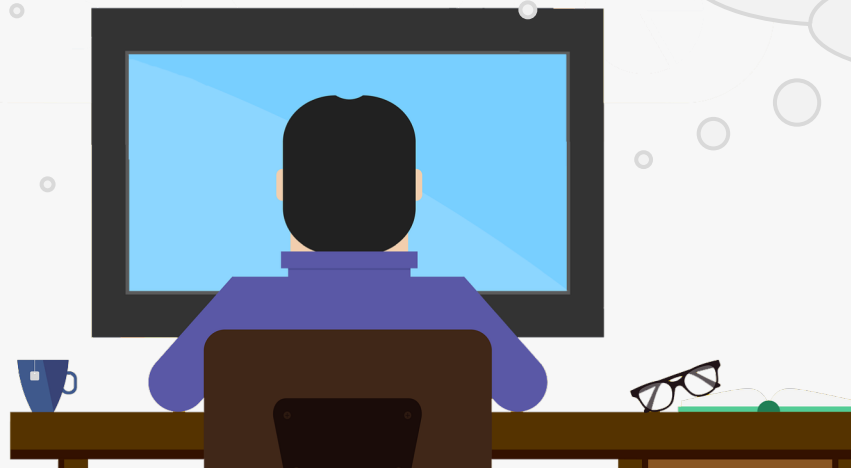
Implementing
new features

Algorithms &
Data structures

Testing

Communication

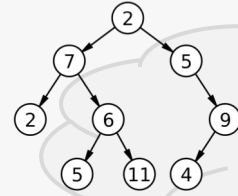
Debugging



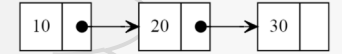
Software Development Expertise?



Implementing new features



Algorithms & Data structures



JUnit 5 Testing *jbehave*



Debugging



Communication





**How to structure all those
expertise-related aspects?**

Which factors influence expertise development over time?



How are experience and expertise related?



Definitions

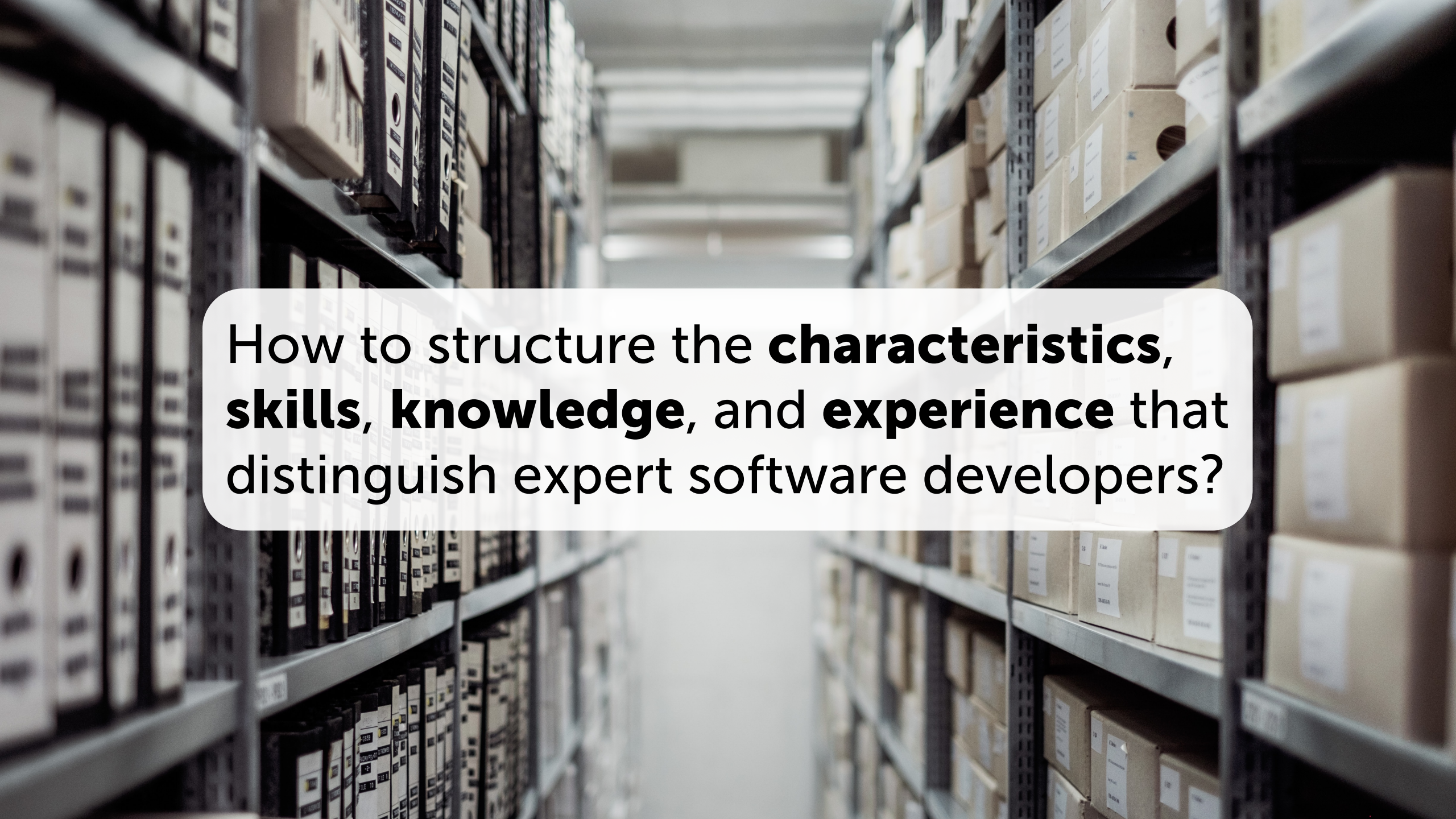
An **expert** is someone “with the special **skill** or **knowledge** representing mastery of a **particular subject**”



Expertise are „the **characteristics, skills, and knowledge** that distinguish experts from novices and less **experienced** people.”



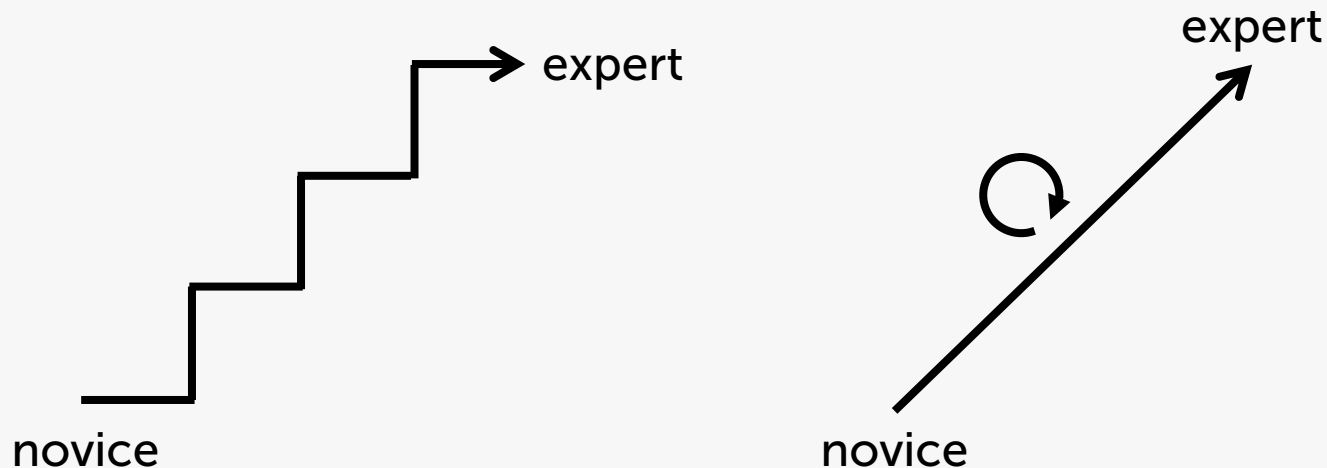
K. Anders Ericsson



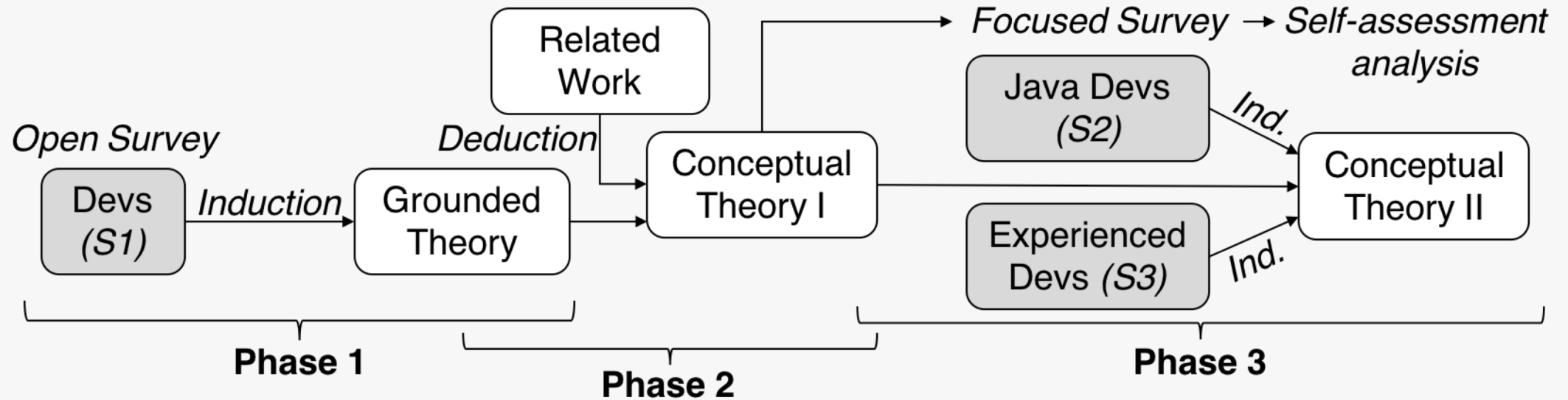
How to structure the **characteristics**, **skills**, **knowledge**, and **experience** that distinguish expert software developers?

Our Expertise Model

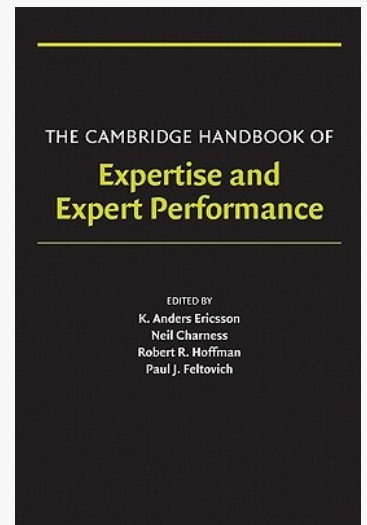
- **Task-specific** (e.g., writing code, debugging, testing)
- Focuses on **individual developers**
- **Process** view (repetition of tasks)
- Notion of **transferable knowledge and experience** from related fields or tasks
- **Continuum** instead of discrete expertise steps



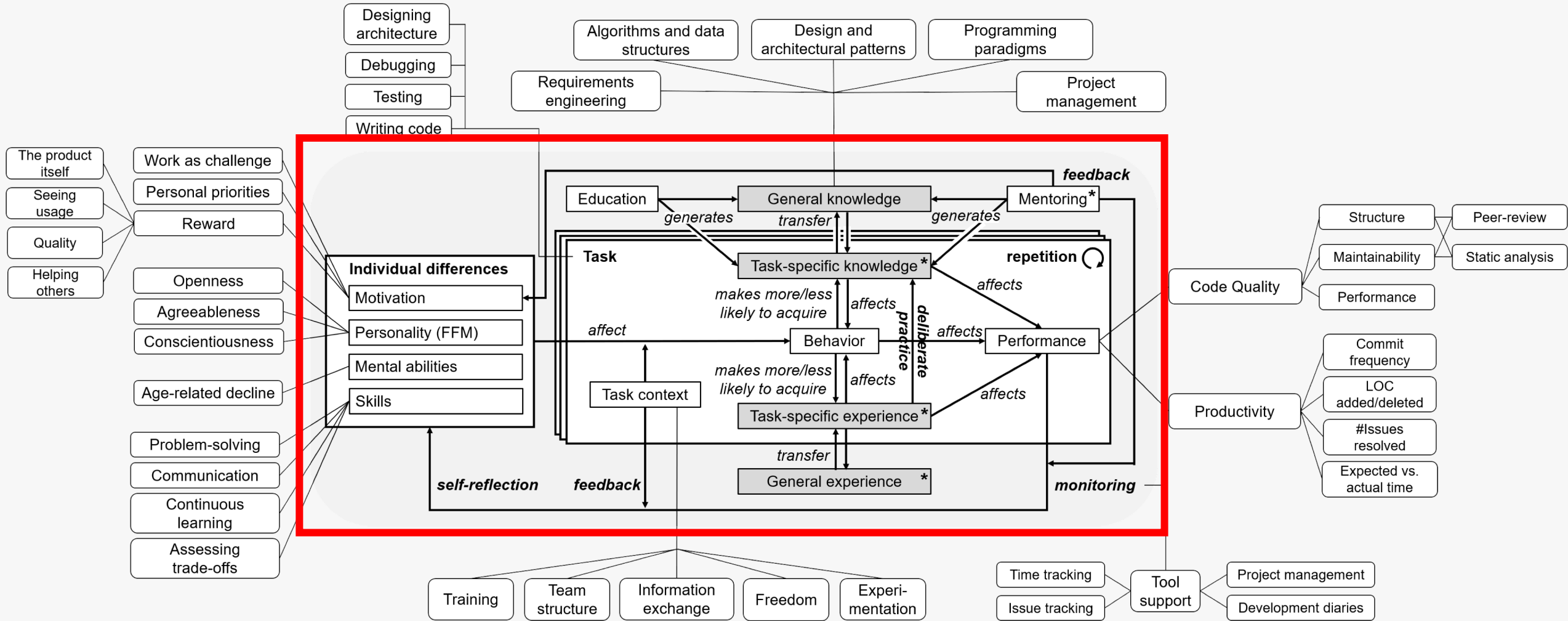
Research Design



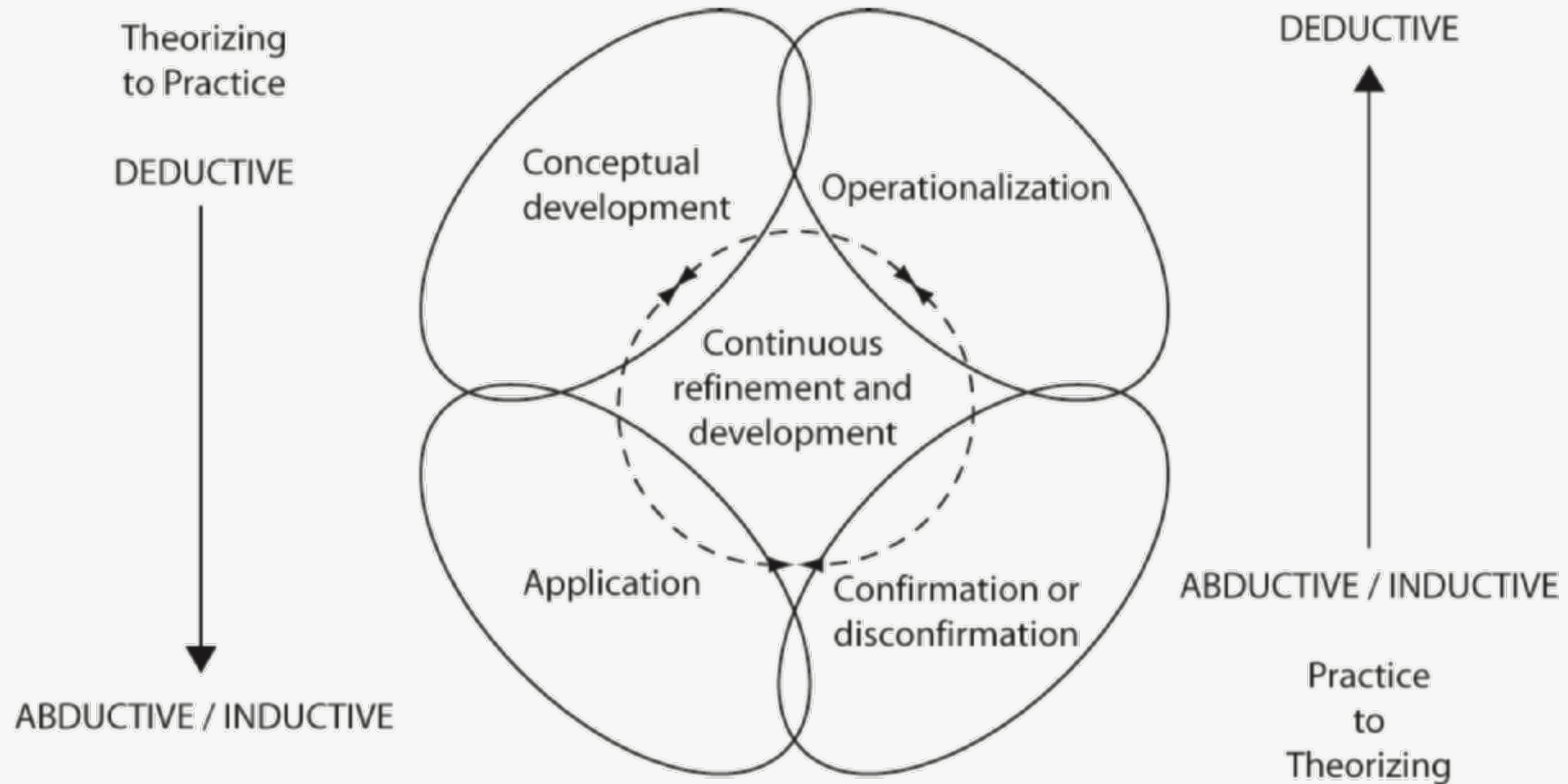
- **Induction:** 335 online survey participants in total
- **Deduction:** Main source "*Cambridge Handbook of Expertise and Expert Performance*"



Final Conceptual Theory

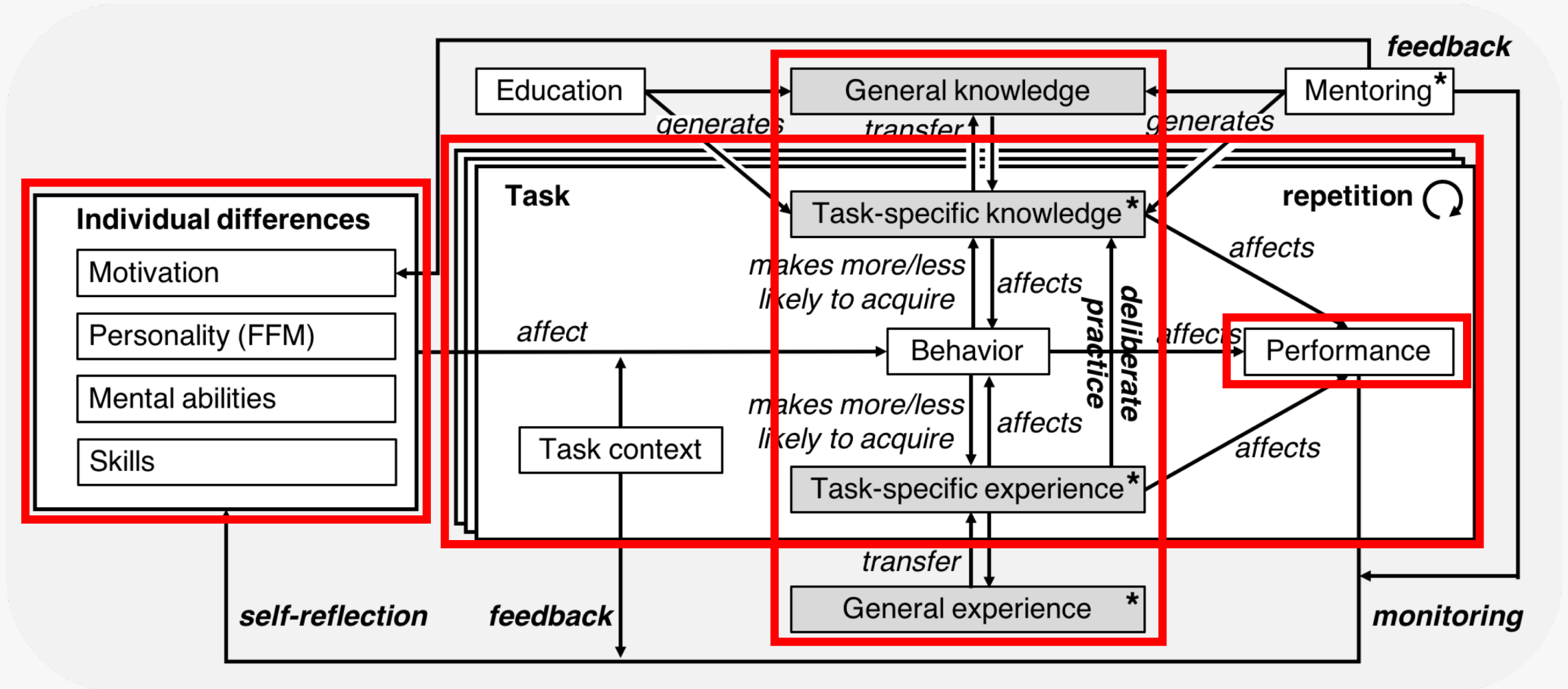


Conceptual Theory?



Building Theories in Software Engineering
Dag I.K. Sjøberg, Tore Dyba, Bente C.D. Anda, and Jo E. Hannay
in *Guide to Advanced Empirical Software Engineering* (2008).

Final Conceptual Theory

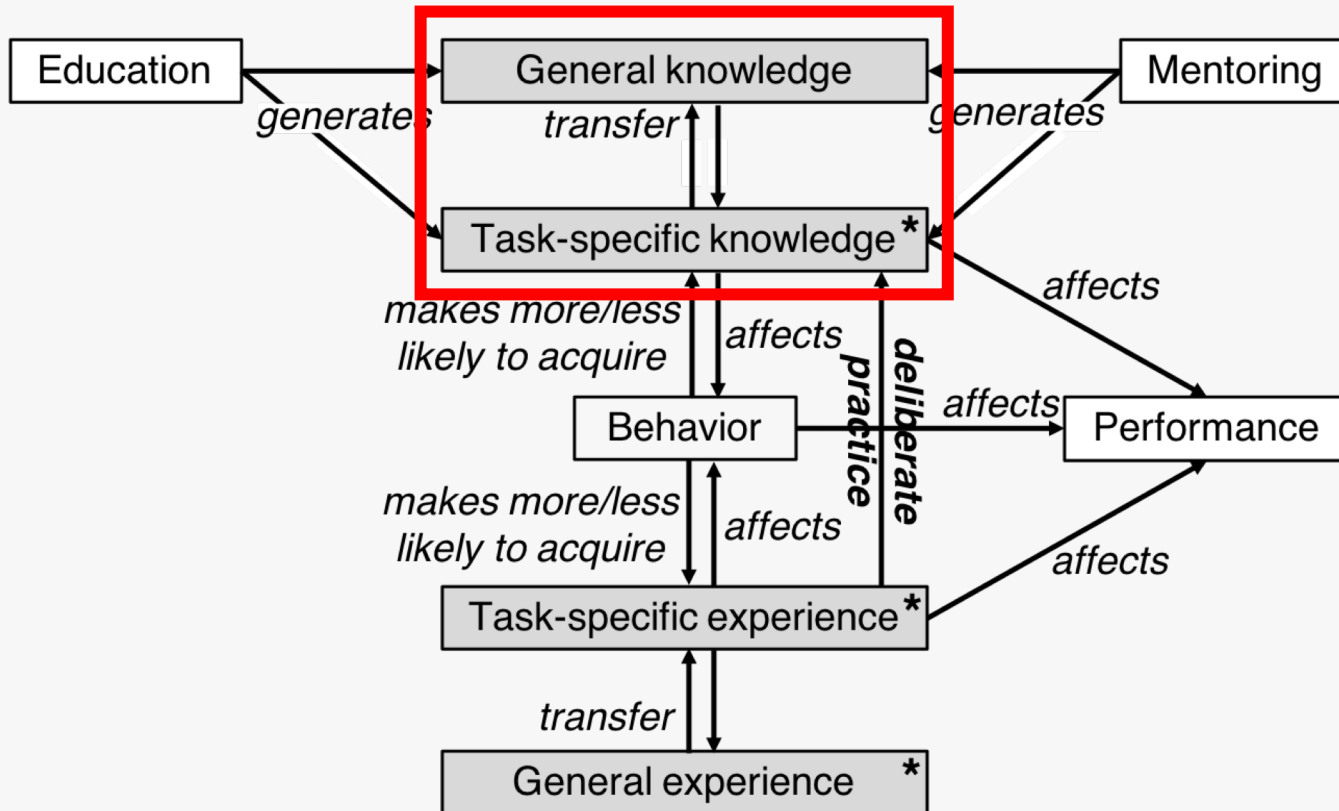
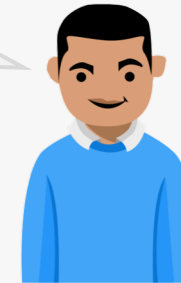


Knowledge

- **Knowledge** is a “*permanent structure of information stored in memory*” (Robillard, 1995)
- Developer’s knowledge base considered (most) important factor influencing **performance** (Curtis, 1984)
- Studies suggest that this knowledge base is “*highly **language dependent***”, but experts also have “*abstract, **transferable knowledge and skills***” (Sonnentag et al., 2006)
- “*Semantic*” vs. “*syntactical*” knowledge (Shneiderman and Mayer, 1978)

Knowledge

Knowledge about “*paradigms [...], data structures, algorithms, computational complexity, and design patterns*”



An “*intimate knowledge of the design and philosophy of the language*”





Context Switch



Question for the Audience I

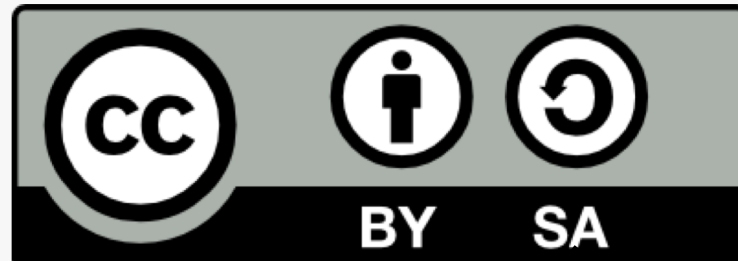
Who admits regularly copying non-trivial code snippets from Stack Overflow?



Question for the Audience II

Who knew that all content on Stack Overflow is licensed under CC BY-SA?

*"You must give **appropriate credit** [...] and indicate if changes were made."*



Attribution

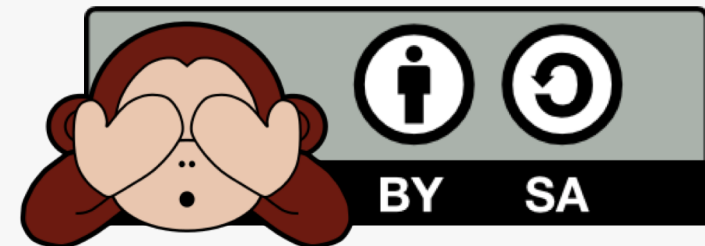
Share-alike

*"If you [...] **build upon** the material, you must **distribute your contributions** under the same license as the original."*

Results from our Online Surveys

- **46%** of the participants admitted copying code from Stack Overflow **without attribution**
- **75%** did **not know** that content on SO is licensed under **CC BY-SA**
- **67%** did **not know** that **attribution is required**

→ **Lack of awareness**



Background



“Well, but these snippets are rather trivial and not protected by copyright.”

- Not all code snippets on Stack Overflow are copyrightable
- “A snippet that is more than one or two lines of standard function calls would typically be creative enough for copyright” [Engelfriet 2016]
- But no “international standard for originality” [Creative Commons 2017b]



889

- Here's what I do:
1. First of all I check what providers are enabled. Some may be disabled on the device, some may be disabled in application manifest.
 2. If any provider is available I start location listeners and timeout timer. It's 20 seconds in my example, may not be enough for GPS so you can enlarge it.
 3. If I get update from location listener I use the provided value. I stop listeners and timer.
 4. If I don't get any updates and timer elapses I have to use last known values.
 5. I grab last known values from available providers and choose the most recent of them.

Here's how I use my class:

```
LocationResult locationResult = new LocationResult(){
    @Override
    public void getLocation(Location location){
        //Got the location!
    }
};
MyLocation myLocation = new MyLocation();
myLocation.getLocation(this, locationResult);
```

And here's MyLocation class:

```
import java.util.Timer;
import java.util.TimerTask;
import android.content.Context;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;

public class MyLocation {
    Timer timer1;
    LocationManager lm;
    LocationResult locationResult;
    boolean gps_enabled=false;
    boolean network_enabled=false;

    public boolean getLocation(Context context, LocationResult result)
    {
        //I use LocationResult callback class to pass location value from MyLocat
        locationResult=result;
        if(lm==null)
            lm = (LocationManager) context.getSystemService(Context.LOCATION_SERV

        //exceptions will be thrown if provider is not permitted.
        try(gps_enabled=lm.isProviderEnabled(LocationManager.GPS_PROVIDER);}catch
        try(network_enabled=lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER

        //don't start listeners if no provider is enabled
        if(!gps_enabled && !network_enabled)
            return false;

        if(gps_enabled)
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locatio
        if(network_enabled)
            lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, loc
```

Somebody may also want to modify my logic. For example if you get update from Network provider don't stop listeners but continue waiting. GPS gives more accurate data so it's worth waiting for it. If timer elapses and you've got update from Network but not from GPS then you can use value provided from Network.

One more approach is to use LocationClient <http://developer.android.com/training/location/retrieve-current.html>. But it requires Google Play Services apk to be installed on user device.

share improve this answer edited Jun 25 '13 at 9:33 answered Jun 30 '10 at 0:07

Fedor
40k ● 9 ● 71 ● 86



```
public class MyLocation {
    Timer timer1;
    LocationManager lm;
    LocationResult locationResult;
    boolean gps_enabled=false;
    boolean network_enabled=false;

    public boolean getLocation(Context context, LocationResult result)
    {
        // Use LocationResult callback class to pass location value from MyLocation to user code.
        locationResult=result;
        if(lm==null)
            lm = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);

        //exceptions will be thrown if provider is not permitted.
        try(gps_enabled=lm.isProviderEnabled(LocationManager.GPS_PROVIDER);}catch(Exception ex){}
        try(network_enabled=lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);}catch(Exception ex){}

        //don't start listeners if no provider is enabled
        if(!gps_enabled && !network_enabled)
            return false;

        if(gps_enabled)
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListenerGps);
        if(network_enabled)
            lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListenerNetwork);
        timer1=new Timer();
        timer1.schedule(new GetLastLocation(), 20000);
        return true;
    }

    LocationListener locationListenerGps = new LocationListener() {
        public void onLocationChanged(Location location) {
            timer1.cancel();
            locationResult.getLocation(location);
            lm.removeUpdates(this);
            lm.removeUpdates(locationListenerNetwork);
        }
        public void onProviderDisabled(String provider) {}
        public void onProviderEnabled(String provider) {}
        public void onStatusChanged(String provider, int status, Bundle extras) {}
    };

    LocationListener locationListenerNetwork = new LocationListener() {
        public void onLocationChanged(Location location) {
            timer1.cancel();
            locationResult.getLocation(location);
            lm.removeUpdates(this);
            lm.removeUpdates(locationListenerGps);
        }
        public void onProviderDisabled(String provider) {}
        public void onProviderEnabled(String provider) {}
        public void onStatusChanged(String provider, int status, Bundle extras) {}
    };

    class GetLastLocation extends TimerTask {
        @Override
        public void run() {
            lm.removeUpdates(locationListenerGps);
            lm.removeUpdates(locationListenerNetwork);

            Location net_loc=null, gps_loc=null;
            if(gps_enabled)
                gps_loc=lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if(network_enabled)
                net_loc=lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);


            //if there are both values use the latest one
            if(gps_loc!=null && net_loc!=null){
                if(gps_loc.getTime()>net_loc.getTime())
                    locationResult.getLocation(gps_loc);
                else
                    locationResult.getLocation(net_loc);
                return;
            }

            if(gps_loc!=null){
                locationResult.getLocation(gps_loc);
                return;
            }
            if(net_loc!=null){
                locationResult.getLocation(net_loc);
                return;
            }
            locationResult.getLocation(null);
        }
    }



    public static abstract class LocationResult{
        public abstract void getLocation(Location location);
    }
}
```



Stack Overflow Code in the OpenJDK

 JDK / JDK-8170860
Get rid of the humanReadableByteCount() method in openjdk/hotspot

Details

Type:	 Bug	Status:	RESOLVED
Priority:	 P2	Resolution:	Fixed
Affects Version/s:	9	Fix Version/s:	9
Component/s:	hotspot		

implement the method `humanReadableByteCount` which body was copied from the Stack Overflow site: <https://stackoverflow.com/a/3758880>

It's just a few lines of code, but it **could cause legal issues.** The method should be either re-implemented or removed.

Besides the potential legal issues, duplicating a code is **not a good practice.**

<https://bugs.openjdk.java.net/browse/JDK-8170860>

... and in Microsoft GitHub Repos

Microsoft / ApplicationInsights-Home

Code Issues 72 Pull requests 0 Projects 0 Wiki Insights

Unclear licensing situation for code in AccountController.cs #328

Open

Microsoft / rDSN

Code Issues 5 Pull requests 1 Projects 0 Wiki Insights

Unclear licensing situation for code in csproj.template.php #209

Open

Microsoft / Windows-universal-samples

Code Issues 42 Pull requests 55 Projects 0 Wiki Insights

Unclear licensing situation for code in BindableFlyout.cs #1070

Open sbaltes opened this issue a day ago · 1 comment

Yes.

11 I put together a simple solution for developers who desire this functionality. It uses an attached property to identify the ItemsSource and the ItemTemplate for a Flyout control. If the developer elects to use a MenuFlyoutItem or something else, it is up to them.

Here's the attached property:

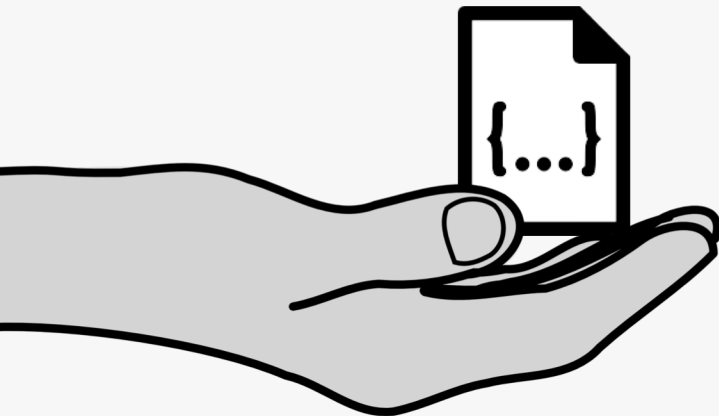
```
public class BindableFlyout : DependencyObject
{
    #region ItemsSource
    public static IEnumerable GetItemsSource(DependencyObject obj)
    {
        return obj.GetValue(ItemsSourceProperty) as IEnumerable;
    }
    public static void SetItemsSource(DependencyObject obj, IEnumerable value)
    {
        obj.SetValue(ItemsSourceProperty, value);
    }
    public static readonly DependencyProperty ItemsSourceProperty =
        DependencyProperty.RegisterAttached("ItemsSource", typeof(IEnumerable),
            typeof(BindableFlyout), new PropertyMetadata(null, ItemsSourceChanged));
    private static void ItemsSourceChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
    {
        Setup(d as Windows.UI.Xaml.Controls.Flyout);
    }
    #endregion

    #region ItemTemplate
    public static DataTemplate GetItemTemplate(DependencyObject obj)
    {
        return (DataTemplate)obj.GetValue(ItemTemplateProperty);
    }
    public static void SetItemTemplate(DependencyObject obj, DataTemplate value)
    {
        obj.SetValue(ItemTemplateProperty, value);
    }
    public static readonly DependencyProperty ItemTemplateProperty =
        DependencyProperty.RegisterAttached("ItemTemplate", typeof(DataTemplate),
            typeof(BindableFlyout), new PropertyMetadata(null, ItemTemplateChanged));
    private static void ItemTemplateChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)
    {
    }
    #endregion
}
```

Implications of Stack Overflow's License

Permissive Licenses

- Permit using the licensed source code in proprietary software **without publishing changes** or the **derived work**
- *Examples:* MIT, Apache, and BSD license families



Copyleft Licenses

- Requires either modifications to the licensed content or the complete derived work to be **published under the same or a compatible license** (share-alike)
- *Examples (weak copyleft):* Mozilla/Eclipse Public Licenses
- *Examples (viral copyleft):* GNU General Public Licenses, Creative Commons Share-Alike Licenses (e.g., **CC BY-SA**)

Enforceability of Copyleft Licenses

- Courts in the US and Europe ruled that open source licenses are **enforceable contracts**
- Authors are able to **sue** when terms such as the share-alike requirement are violated:
 - **Interdict distribution** of derived work
 - **Claim monetary damages**
- USA: DMCA takedown notices for allegedly infringed copyright
 - Example: <https://github.com/github/dmca>
- Risk in mergers and acquisitions of companies
 - Example: FSF vs. Cisco lawsuit





Research Question



Question:

How **frequently** is code from Stack Overflow posts used in public GitHub projects **without** the required **attribution**?

Approach:

Triangulate an estimate for the attribution ratio using three different methods.

Attribution



Attribution ratio:

- Method 1 (regular expressions): 23 %
- Method 2 (code clone detector): 24 %
- Method 3 (exact matches): 8 %

Conservative estimate:

- **Attribution ratio \leq 25%**

Share-alike



Only **2%** of all analyzed repositories (all methods) containing code from Stack Overflow **attributed** its source and used a **compatible license** (not CC BY-SA, but GPL 3.0).

SPDX license name	Number of repos containing a SO code snippet clone that was:	
	unattributed (<i>n</i> = 2,962)	attributed (<i>n</i> = 329)
Apache-2.0	921 (31.1%)	99 (30.1%)
MIT	621 (21.0%)	72 (21.9%)
GPL-3.0	435 (14.7%)	60 (18.2%)
GPL-2.0	284 (9.6%)	21 (6.4%)
BSD-3-Clause	82 (2.8%)	9 (2.7%)

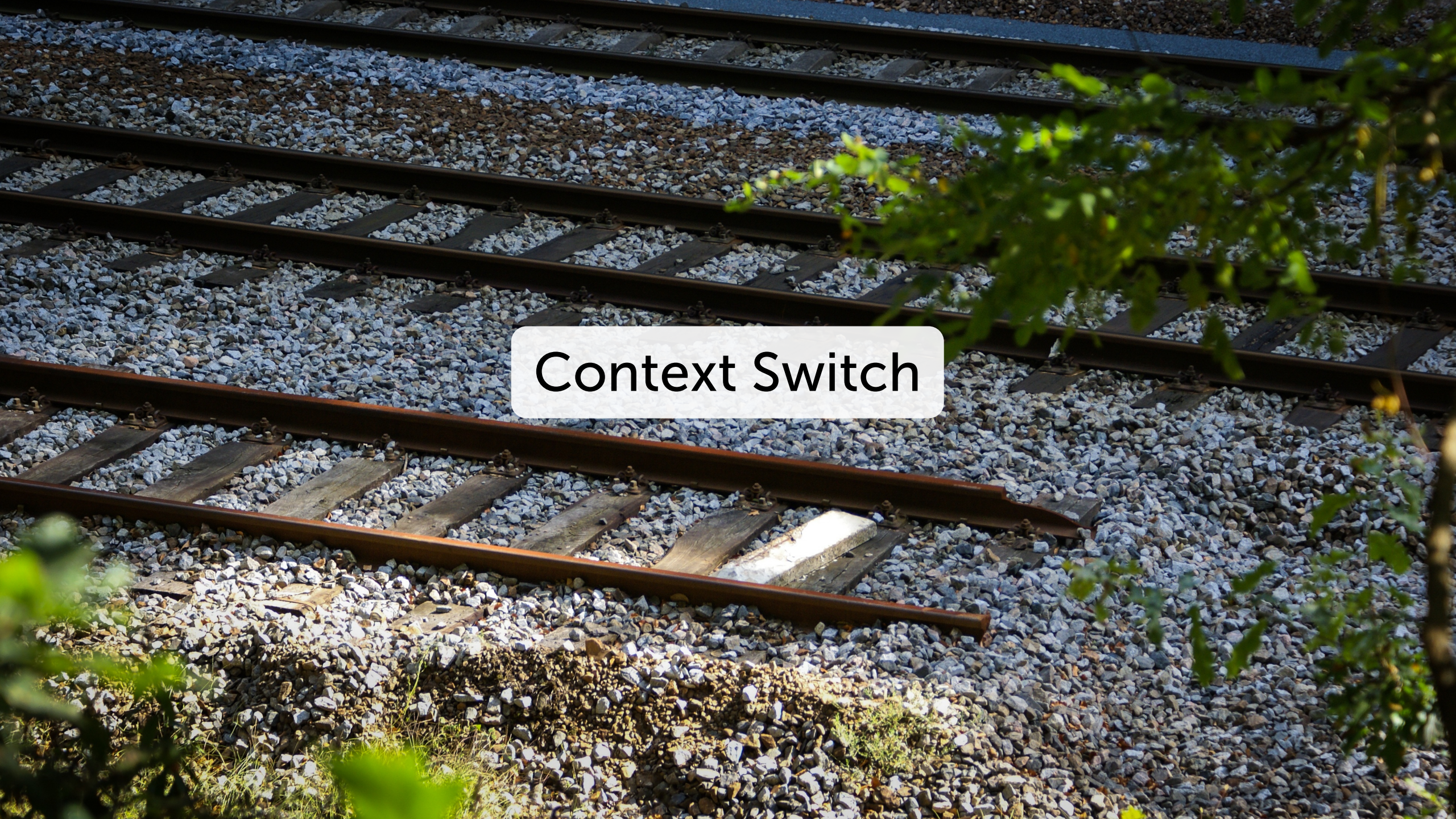
Method 1

SPDX license name	Number of repos containing a SO code snippet clone that was:	
	unattributed (<i>n</i> = 144)	attributed (<i>n</i> = 55)
None	56 (38.9%)	18 (32.7%)
Apache-2.0	33 (22.9%)	15 (27.3%)
GPL-3.0	17 (11.8%)	6 (10.9%)
MIT	6 (4.2%)	4 (7.3%)
GPL-2.0	4 (2.8%)	2 (3.6%)

Method 2

SPDX license name	Number of repos containing a SO code snippet clone that was:	
	unattributed (<i>n</i> = 1,169)	attributed (<i>n</i> = 163)
Apache-2.0	353 (30.2%)	36 (37.4%)
MIT	239 (20.4%)	25 (15.3%)
GPL-3.0	211 (18.0%)	19 (11.7%)
None	153 (13.1%)	61 (37.4%)
GPL-2.0	89 (7.61%)	8 (4.9%)

Method 3

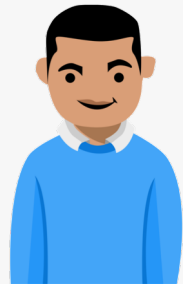


Context Switch

Experience

- Many participants mentioned not only the **quantity**, but also the **quality of experience**

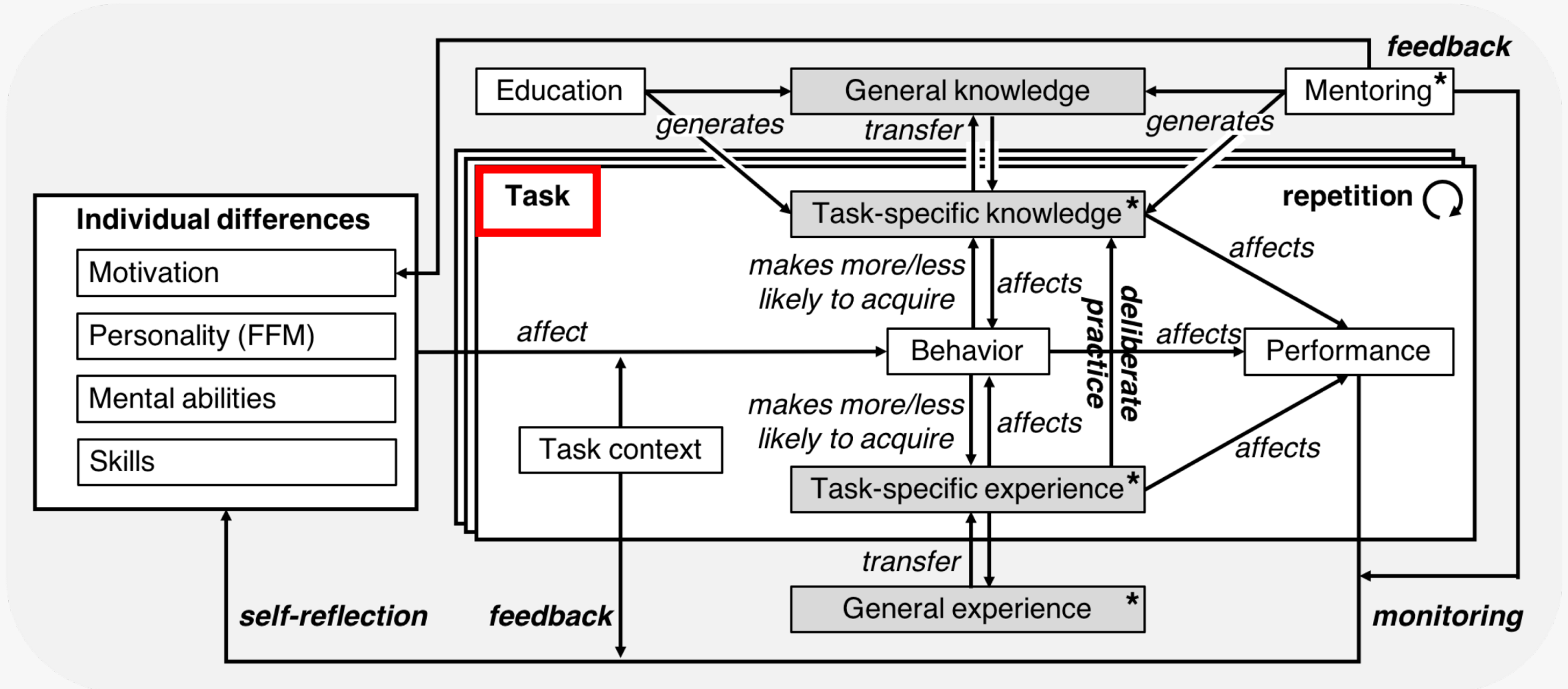
Having built „everything from small projects to enterprise projects“



Having shipped „a significant amount of code to production or to a customer“



Final Conceptual Theory



Tasks

- Asked participants to name the **three most important tasks** that a software development expert should be good at
- Most frequently mentioned:
 1. Designing a software architecture
 2. Writing source code
 3. Analyzing and understanding requirements
- Other mentioned tasks: testing, communicating, debugging

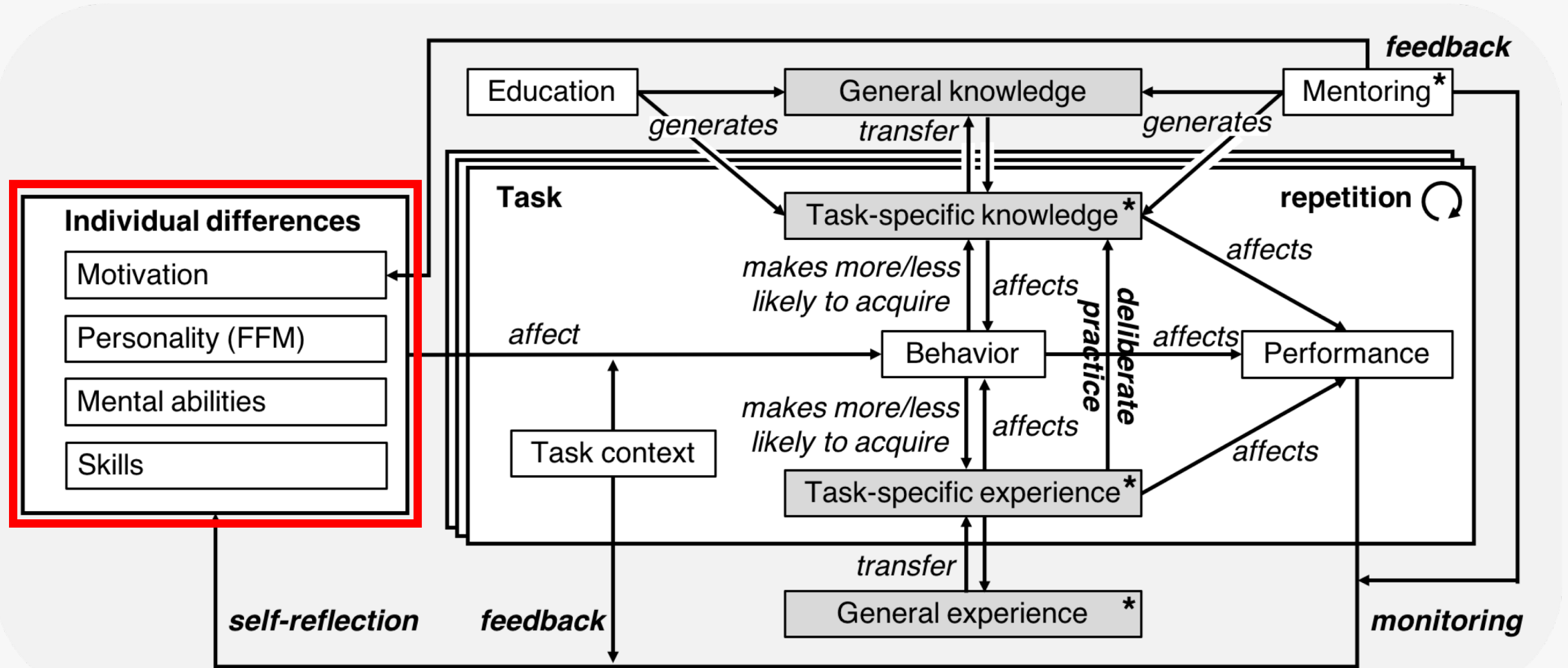
“Architecting the software in a way that allows flexibility in project requirements and future applications of the components”



Which factors influence expertise development over time?



Final Conceptual Theory



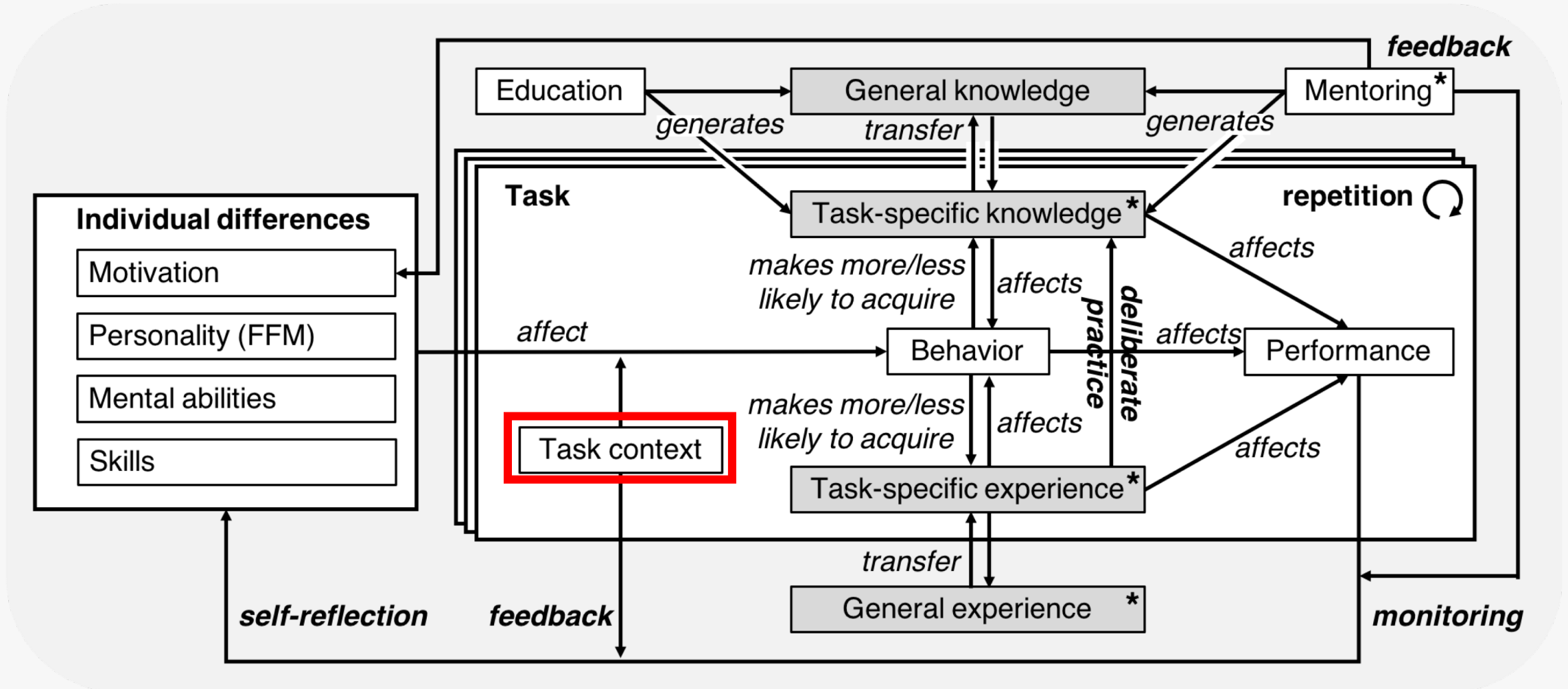
Individual Differences: Motivation

- Related work describes how **individual differences** affect expertise development
- Mental abilities and personality are relatively stable
- **Motivation can change** over time
- Many participants **intrinsically motivated**:
 - Problem solving
 - Seeing a high-quality solution
 - Creating something new
 - Helping others

*"The initial design is fun, but what really is more rewarding is **refactoring**."*

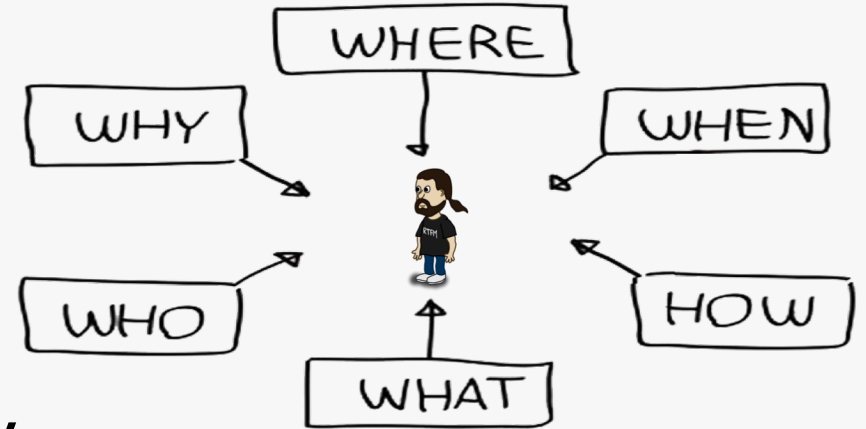


Final Conceptual Theory

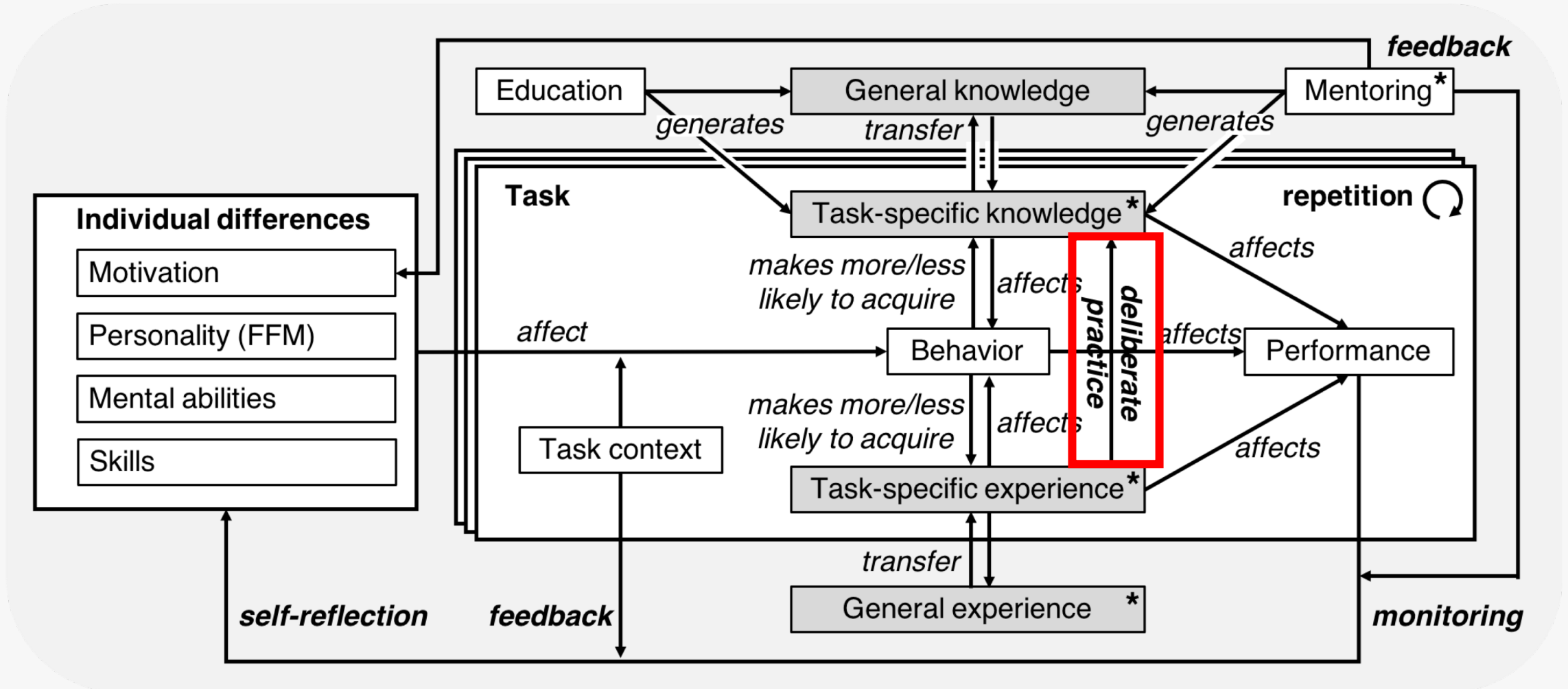


Task Context

- **Work environment**
(office, coworkers, customers etc.)
- **Project constraints**
(external dependencies, time, etc.)
- Can either **foster or hinder** expertise dev.
- We asked: *What can employers do?*
 1. **Encourage learning**
(training courses, library, monetary incentives)
 2. **Encourage experimentation**
(side projects, being open to new ideas/technologies)
 3. **Improve information exchange**
(facilitate meetings, rotating between teams/projects)
 4. **Grant freedom**
(less time pressure)



Final Conceptual Theory



Deliberate Practice



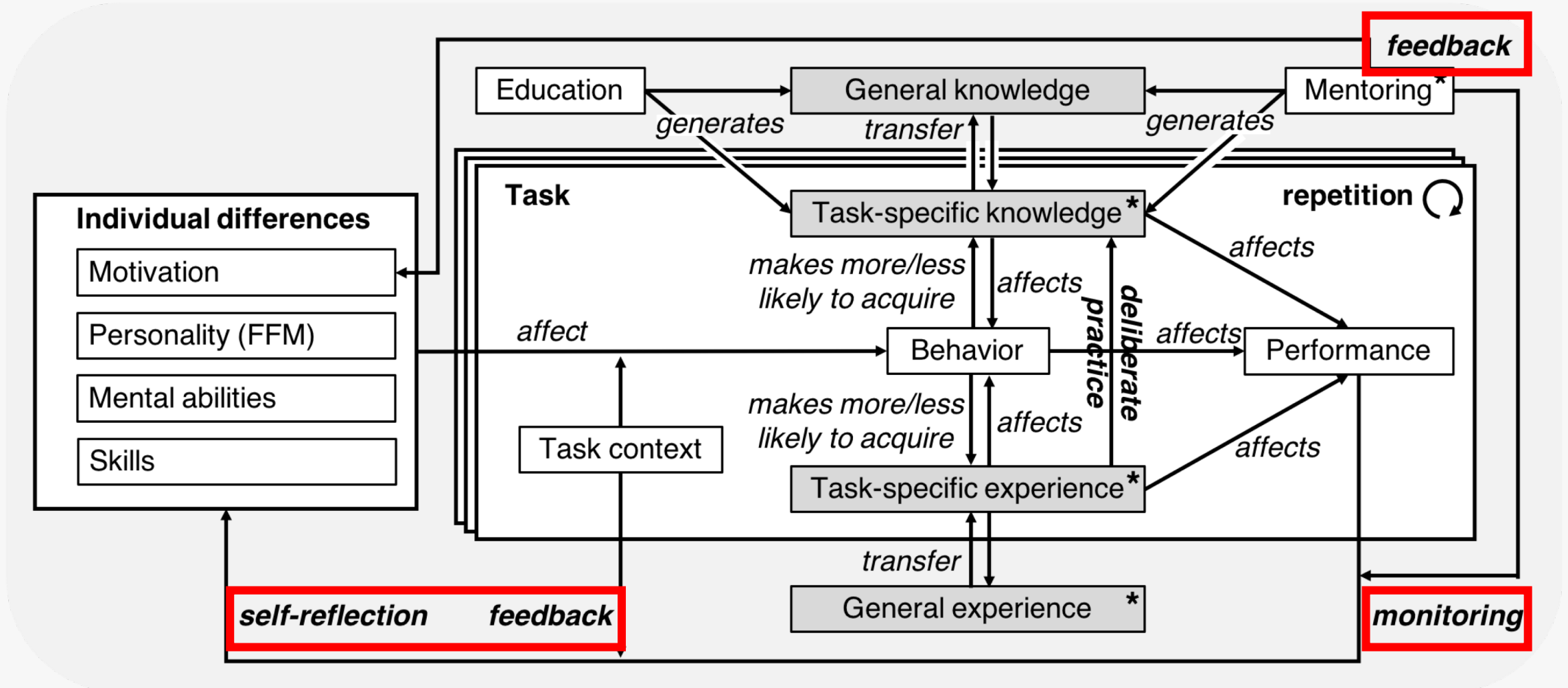
- Having **more experience** does not automatically lead to **better performance** (Ericsson et al., 1993)
- Performance may even **decrease** over time (Feltovich, 2006)
- Length of experience only weak correlate of job performance (Ericsson, 2006)
- Deliberate practice: „***Prolonged efforts to improve performance while negotiating motivational and external constraints***“ (Ericsson et al., 1993)

Deliberate Practice: Self-Reflection

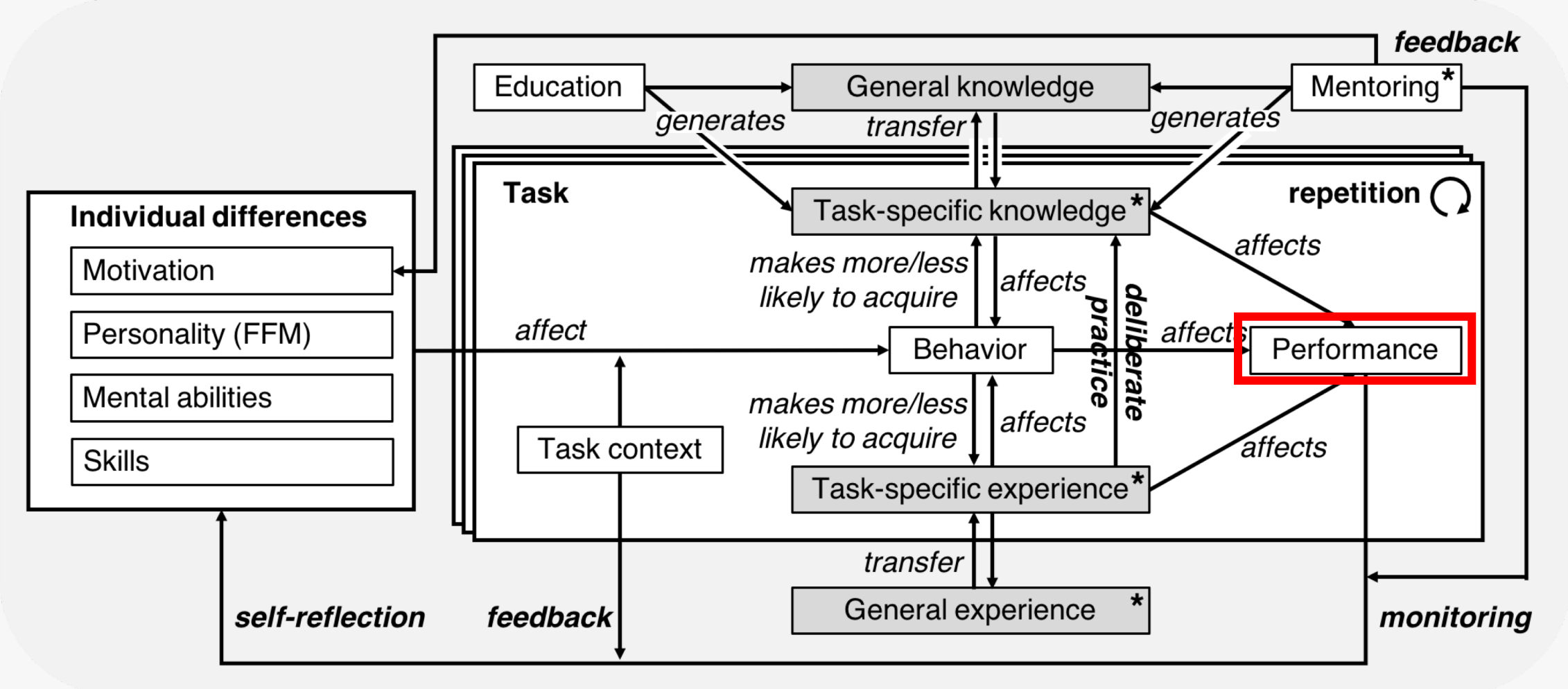


- **(Self-)reflection** and **feedback** important to **monitor** progress towards goal achievement (Locke and Latham, 1990)
- *“[T]he more **channels of accurate and helpful feedback** we have access to, the better we are likely to perform.”*
(Tourish and Hargie, 2003)
- **Mentors**, teachers, and peers are an important sources for feedback

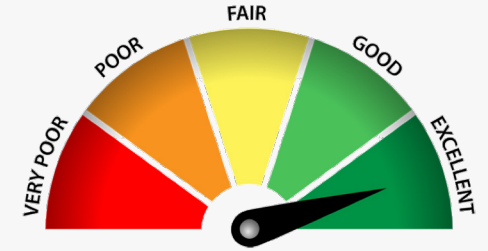
Final Conceptual Theory



Final Conceptual Theory



Performance



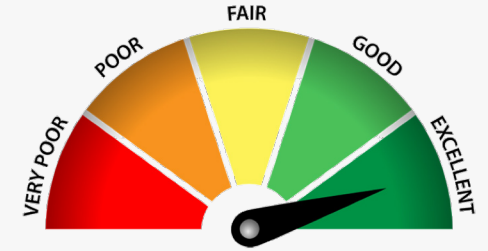
Scope of this work:

- We do **not** treat performance as a **dependent variable** that we try to explain for individual tasks
- We consider different **performance monitoring** approaches to be a means for feedback and self-reflection

Long-term goal:

- Build **variance theory** for explaining and predicting the development of expertise

Performance



- Participants described different **properties of expert's source code** (well-structured, readable, maintainable, etc.)

*„Everyone can write [...] code which a machine can read and process but the key lies in writing concise and understandable code which [...] **people who have never used that piece of code before [can read].**“*



Performance Decline

- Goal: Identify factors **hindering** expertise development
- **41.5%** of participants observed a **significant performance decline** over time (for themselves or others)
- Reasons:
 - Demotivation
 - Changes in the work environment
 - Age-related decline
 - Changes in attitude
 - Shifting towards other tasks

*“I perceived an **increasing procrastination** in me and in my colleagues, by **working on the same tasks** over a relatively long time [...] **without innovation and environment changes.**”*



Age-Related Performance Decline

*“For myself, it’s mostly the effects of aging on the brain. At age 66, **I can’t hold as much information short-term memory**, for example. [...] I can compensate for a lot of that by writing simpler functions with clean interfaces. The results are still good, but **my productivity is much slower than when I was younger.**”*



software architect, age 66

*“Programming ability is based on **desire to achieve**. In the early years, it is a sort of **competition**. [...] I found that I lost a significant amount of my focus as I became 40, and started **using drugs such as ritalin** to enhance my abilities. This is pretty common among older programmers.”*



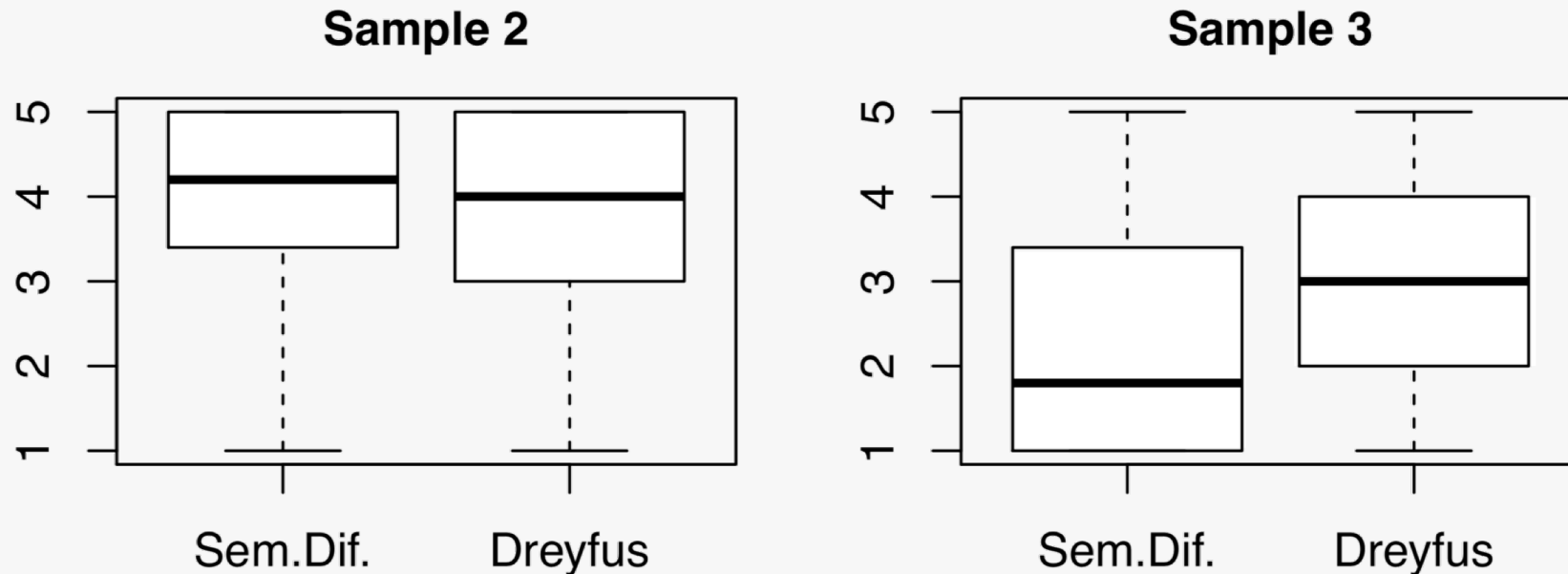
software developer, age 60

How are experience and expertise related?



Experience vs. Expertise

- Self-assessment with **semantic differential** (novice to expert) and **Dreyfus expertise model**
- More experienced developers **adjusted** their ratings when context was provided, less experienced not





Takeaways

Summary for Developers

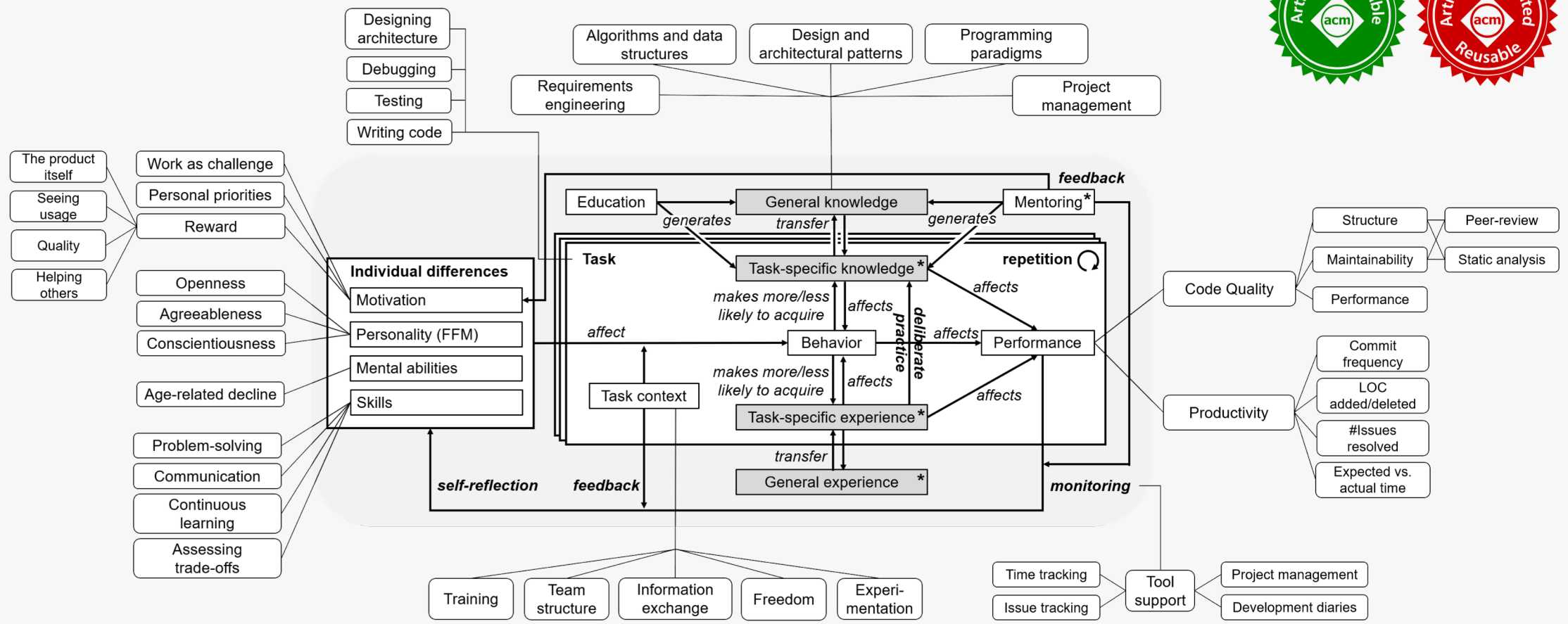
- See which **attributes** other developers assign to experts
- Learn which **behaviors** may lead to becoming a better software developer:
 - Deliberate practice
 - Have challenging goals
 - Build or maintain a supportive work environment (also for others)
 - Ask for feedback from peers
 - Reflect about what one knows and what not



Summary for Employers

- Learn what **(de)motivates** their employees:
 - Main motivation: problem solving
 - Main demotivation: non-challenging work
- Ideas on how to build supportive work environment **supporting self-improvement** of staff:
 - Good mix of continuity and change in software development process
 - Communicate clear visions, directions, and goals
 - Reward high-quality work wherever possible
 - Revisit information sharing in company





Dr. Sebastian Baltes
 @s_baltes

expertise.sbaltes.com
Data and scripts available on Zenodo