

Developer Experience...

Central to *Enterprise* Success

What is "Developer Experience" (DevEx)?

DevEx as Disaster

Common examples

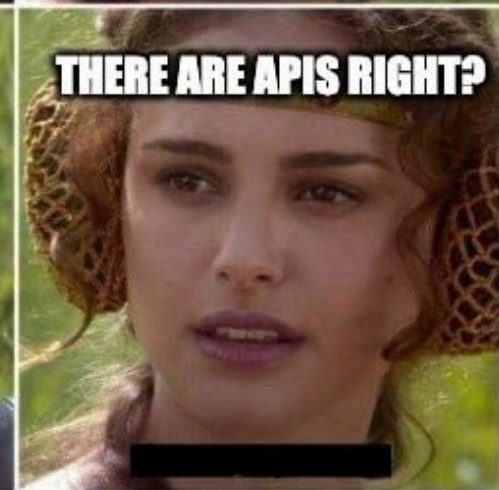
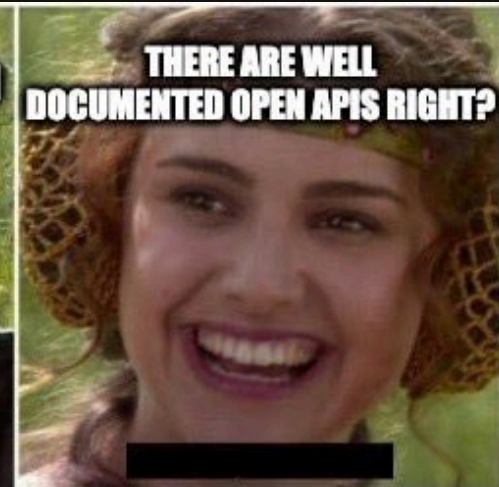
- Poorly documented features (or bugs)



DevEx as Disaster

Common examples

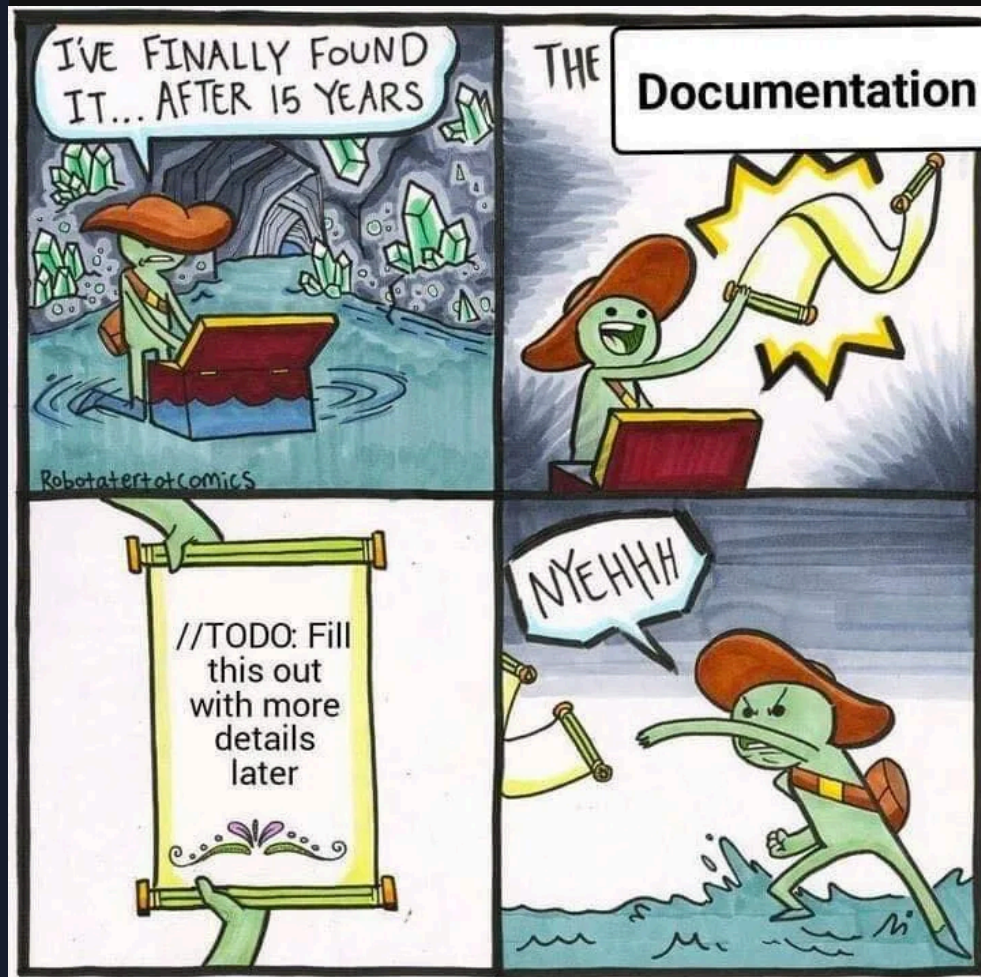
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)



DevEx as Disaster

Common examples

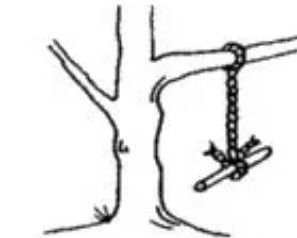
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- Downloading documentation... as a PDF, or access-gated



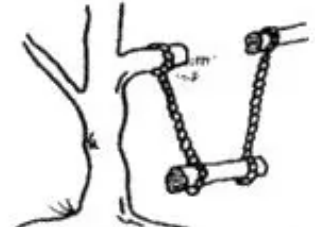
DevEx as Disaster

Common examples

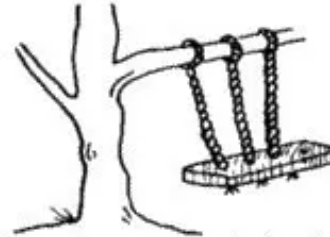
- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- Downloading documentation... as a PDF, or access-gated
- Missing examples... of *anything*



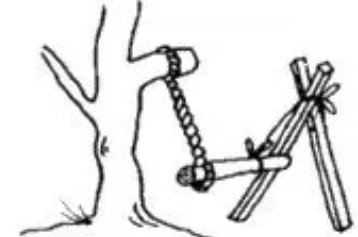
What the user asked for



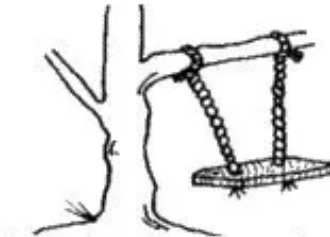
How the analyst saw it



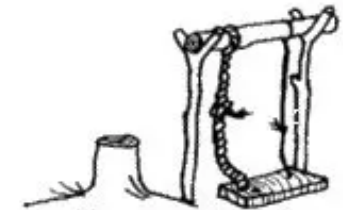
How the system was designed



As the programmer wrote it



What the user really wanted



How it actually works

DevEx as Disaster

Common examples

- Poorly documented features (or bugs)
- Missing OpenAPI spec (or even APIs)
- Downloading documentation... as a PDF, or access-gated
- Missing examples... of *anything*
- “CI as Magic 8-Ball”



Ramiro Berrelleza 
@rberrelleza

Long time ago, in a galaxy far away, I worked at a team where our CI environment was so different from local or production, that the only realistic option way to validate a change was in prod. So we would commit the change, rerun CI jobs until they were green, deploy to prod, and then monitor the logs for about 1 hour. If no major errors were logged after that you were good to go 😊

12:39 AM · Aug 3, 2024

Yale School of Art
1156 Chapel Street, POB 208339
New Haven, Connecticut, 06520-8339

YALE SCHOOL OF ART

- Home
- About the School
- Apply to the School
- Exhibitions
- Publications
- News
- Public Events

Pause animations 

This website exists as an ongoing collaborative experiment in digital publishing and information sharing. Because this website functions as a wiki, all members of the School of Art community—graduate students, faculty, staff, and alumni—have the ability to add new content and pages, and to edit most of the site's existing content.

Content is the property of its various authors. When you contribute to this site, you agree to abide by Yale University academic and network use policy, and to act as a responsible member of our community.

The Yale School of Art is a **graduate school** that confers MFAs in Graphic Design, Painting/Printmaking, Photography, and Sculpture; and offers undergraduate-level art courses to Yale College students. Our website exists as an **ongoing collaborative experiment** in digital publishing and information sharing. It functions as a wiki—all members of the School of Art community have the ability to add new, and edit most existing content.

[Editor details](#)

QUICK LINKS 

ON THIS PAGE


- [HAPPENING AT SOA](#)
- [COMMUNITY BULLETIN BOARD](#)
- [CALENDARS & NEWSLETTERS](#)

HAPPENING AT SOA

Visitor: Log in
[Edit this page](#)

[Developing Fall 2024 Visiting Artist lecture schedule here >](#)


```
git push heroku main
```

 Deploy to Heroku



Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](#)

Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452 \[cs.SE\]](#)
(or [arXiv:1312.1452v1 \[cs.SE\]](#) for this version)
<https://doi.org/10.48550/arXiv.1312.1452>

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition," 2012 International Conference on Software and System Process (ICSSP), Zurich, Switzerland, 2012.



[Submitted on 5 Dec 2013]

Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](#)Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452 \[cs.SE\]](#)
 (or [arXiv:1312.1452v1 \[cs.SE\]](#) for this version)
<https://doi.org/10.48550/arXiv.1312.1452>

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

"New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments."

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition". 2012."



[Submitted on 5 Dec 2013]

Developer Experience: Concept and Definition

Fabian Fagerholm, Jürgen Münch

New ways of working such as globally distributed development or the integration of self-motivated external developers into software ecosystems will require a better and more comprehensive understanding of developers' feelings, perceptions, motivations and identification with their tasks in their respective project environments. User experience is a concept that captures how persons feel about products, systems and services. It evolved from disciplines such as interaction design and usability to a much richer scope that includes feelings, motivations, and satisfaction. Similarly, developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance. This article motivates the importance of developer experience, sketches related approaches from other domains, proposes a definition of developer experience that is derived from similar concepts in other domains, describes an ongoing empirical study to better understand developer experience, and finally gives an outlook on planned future research activities.

Comments: 5 pages. The final publication is available at [this http URL](#)Subjects: **Software Engineering (cs.SE)**

Cite as: [arXiv:1312.1452 \[cs.SE\]](#)
 (or [arXiv:1312.1452v1 \[cs.SE\]](#) for this version)
<https://doi.org/10.48550/arXiv.1312.1452>

Journal reference: Proceedings of the International Conference on Software and System Process (ICSSP 2012), pages 73–77, Zurich, Switzerland, June 2–3 2012

DevEx isn't new

"...developer experience could be defined as a means for capturing how developers think and feel about their activities within their working environments, with the assumption that an improvement of the developer experience has positive impacts on characteristics such as sustained team and project performance."

REF: F. Fagerholm and J. Münch, "Developer experience: Concept and definition. 2012."

Jeremy Meiss

Director, DevEx & DevRel

OneStream Software

DevOpsDays Kansas City Organizer



A working definition of DevEx

_"...the **journey** of developers as they learn and deploy technology, which if successful, focuses on eliminating obstacles that hinder a developer or practitioner from achieving success in their endeavors."

-*Jessica West, Co-Founder, DevEx Institute*

Point of clarification

- "DevEx" by default focuses on "developer"
- View "DevEx" as a whole of the lifecycle

DevEx

A Good Developer Experience





Distinguishing DevEx from other Concepts

Distinguishing DevEx from other Concepts

User Experience (UX)

- **Focus:** *prioritizing the end users usability and overall experience; aim to make software intuitive, easy to use, and enjoyable to interact with.*
- **Context:** *involves user research, wireframes, testing product to optimize user satisfaction.*
- **Differs from DevEx:** *_DevEx focuses on making tools, processes, and environments that devs use efficient and pleasant.*

Developer Productivity

- **Focus:** *measured in terms of output, with an emphasis on efficiency and performance.*
- **Context:** *metrics like "time to release", "number of pull requests", or "deployment frequency".*
- **Differs from DevEx:** *they don't capture the full experience of developers, while DevEx encompasses efficiency, the satisfaction, well-being, and support structure of devs.*

Distinguishing DevEx from other Concepts

Developer Experience (DevEx)

- **Focus:** *holistic view encompassing all aspects of the developer journey (usability, efficiency, satisfaction, etc.)*
- **Unique:** *integrates elements of UX and productivity, but with a broader scope of psychological safety, community, and feedback loops.*

BIT EJTERS
BAINCE.



FROM TEXT EDITORS TO CLOUD, IDE:
THE EVOLUTION OF DEVELOPER CE



FOLLOD ES
PECOLIP

Evolution of the IDE

Early text editors

USER FRIENDLY by Illiad



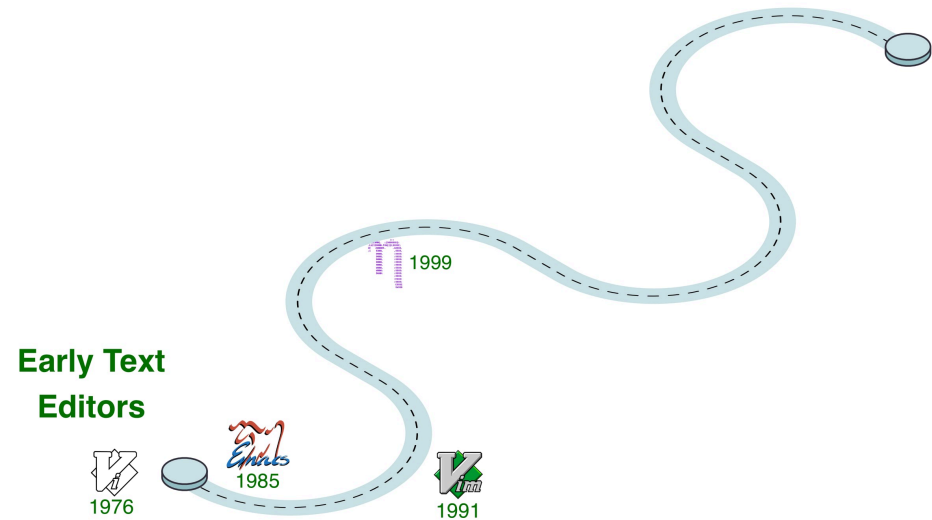
REF: O'Reilly "Learning the vi and Vim Editors"

Evolution of the IDE

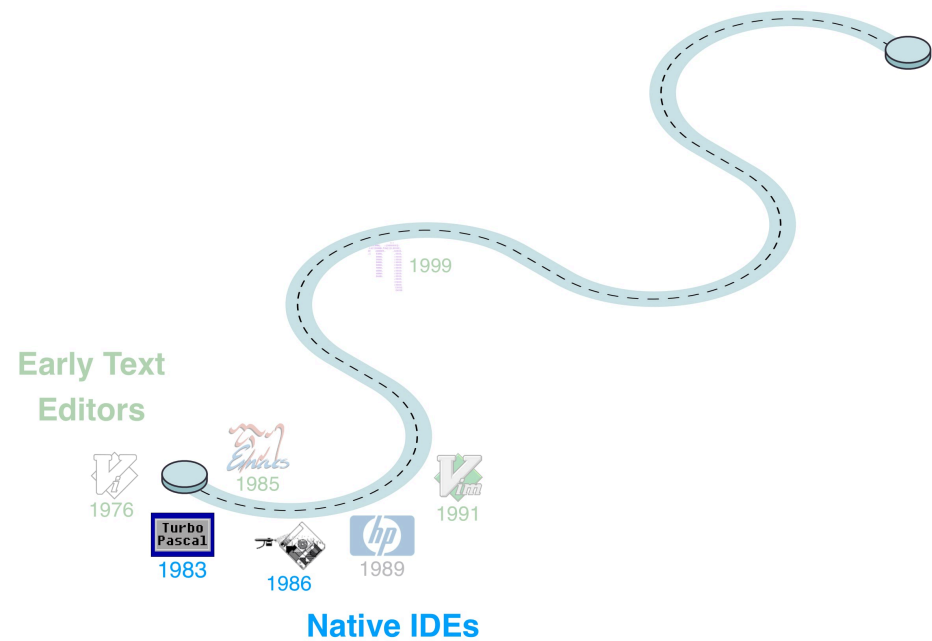
Early text editors

- 1976: Vi
- 1985: Emacs
- 1991: Vim
- 1999: nano

EVOLUTION OF THE IDE



EVOLUTION OF THE IDE

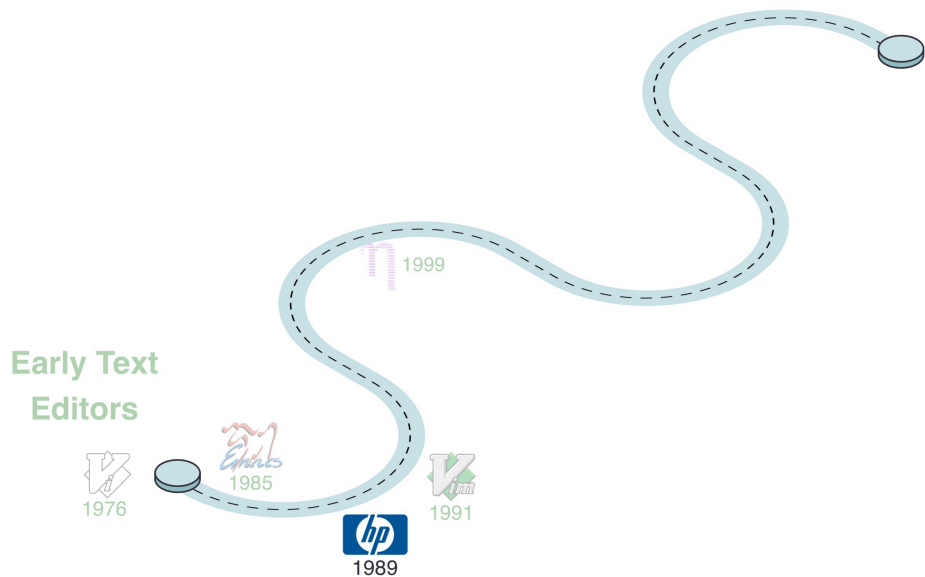


Evolution of the IDE

Native IDEs in the 1980s

- 1983: Turbo Pascal
- 1986: Apple's Macintosh Programmer's Workshop

EVOLUTION OF THE IDE



Evolution of the IDE

First plug-in IDE

HP Softbench

Evolution of the IDE

First plug-in IDE

HP Softbench



Evolution of the IDE

First plug-in IDE

HP Softbench

REF: HP Journal, June 1990 edition

The HP SoftBench Environment: An Architecture for a New Generation of Software Tools

The HP SoftBench product improves programmer productivity by integrating software development tools into a single unified environment, allowing the program developer to concentrate on tasks rather than tools.

by Martin R. Cagan

THE HP SOFTBENCH PRODUCT is an integrated software development environment designed to facilitate rapid, interactive program construction, test, and maintenance in a distributed computing environment.

The HP SoftBench environment provides an architecture for integrating various CASE (computer-aided software engineering) tools. Many of the tools most often needed—program editor, static analyzer, program debugger, program builder, and mail—are included in the HP SoftBench product. Another HP SoftBench component, the HP Encapsulator, makes it possible to integrate other existing tools into the HP SoftBench environment and to tailor the environment to a specific software development process. Fig.

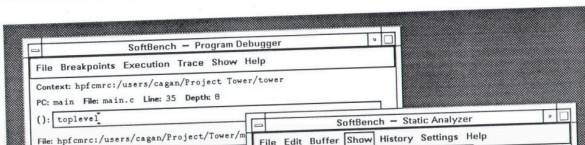
1 illustrates the HP SoftBench user interface.

This article describes the HP SoftBench tool integration architecture. The HP SoftBench program editor, static analyzer, program debugger, program builder, and mail are described in the article on page 48. The HP Encapsulator is described in the article on page 59.

Design Objectives

The overall goal of the HP SoftBench product is to improve the productivity of programmers doing software development, testing, and maintenance. To achieve this goal, the following objectives were defined for the tool integration architecture:

(continued on page 38)





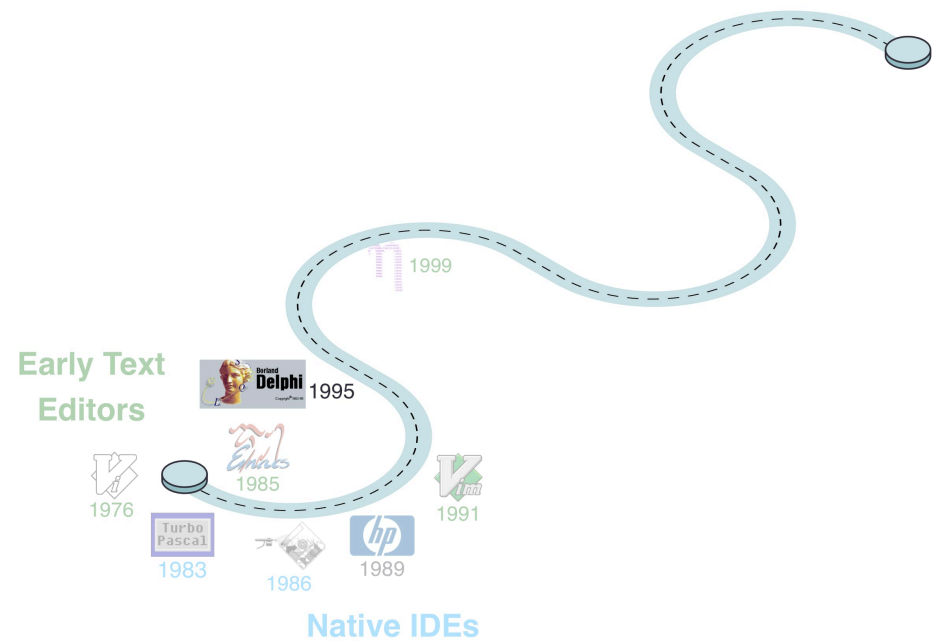
Evolution of the IDE

Early Reviews

"...the use of an IDE was not well received by developers since it would fence in their creativity."

REF: *Computerwoche* ("Computer Week", German counterpart of American magazine *Computer World*), 1995.

EVOLUTION OF THE IDE



Evolution of the IDE

Cross-platform in the 1990s

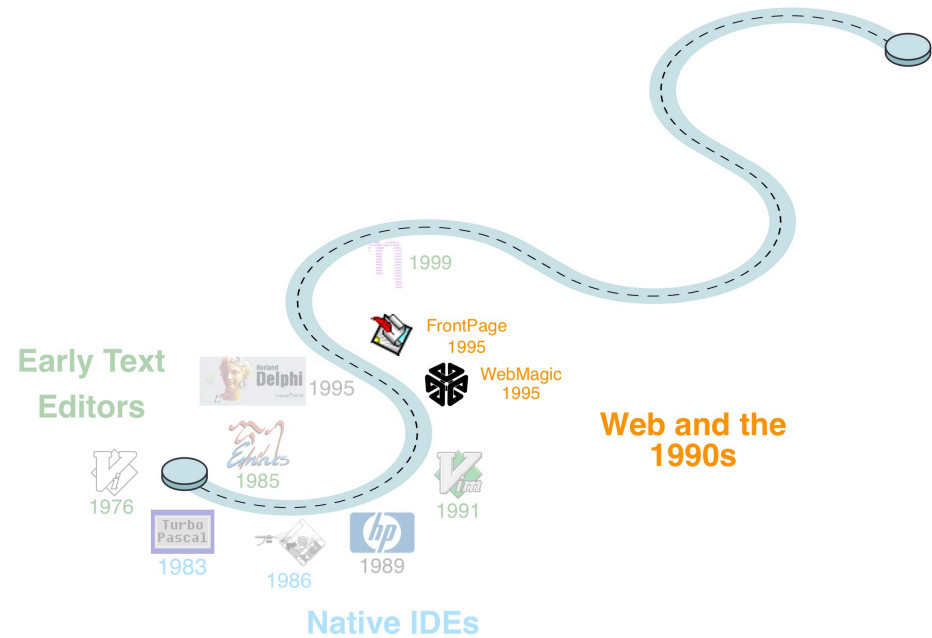
1995: Borland Delphi

Evolution of the IDE

The Web and the 1990s

- 1995: SGI WebMagic
- 1995: Microsoft FrontPage

EVOLUTION OF THE IDE



Evolution of the IDE

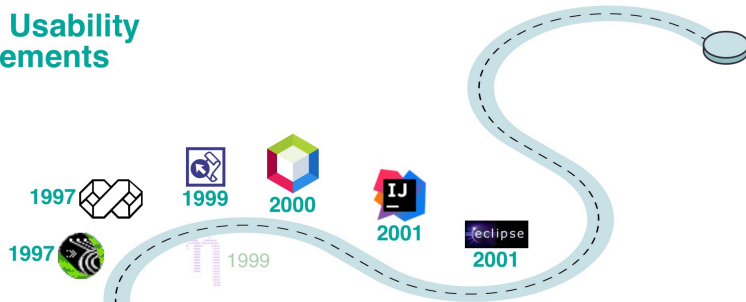
Features & Usability

Late 1990s to 2000s

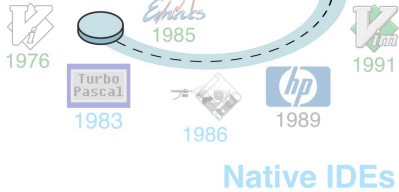
- 1997: Macromedia Dreamweaver
- 1997: Netscape Composer
- 1997: Microsoft Visual Studio
- 1999: Microsoft FrontPage 2000
- 2000: NetBeans
- 2001: IntelliJ IDEA
- 2001: Eclipse IDE
- 2002: Microsoft Visual Studio .NET

EVOLUTION OF THE IDE

Feature & Usability Enhancements



Early Text Editors



Web and the 1990s



Native IDEs

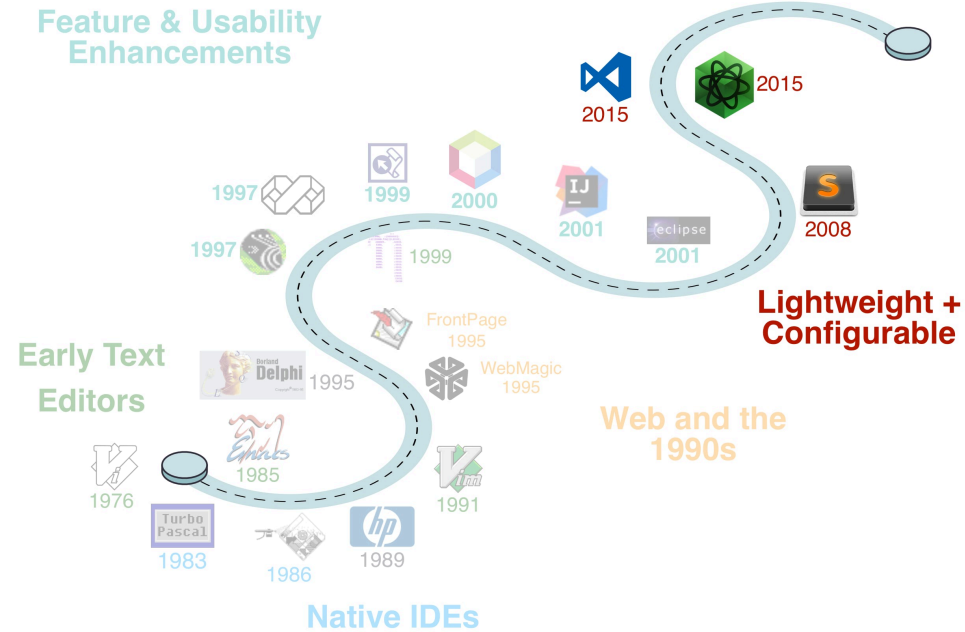
Evolution of the IDE

Lightweight & Configurable

2010s to Now

- 2008: Sublime Text
- 2015: Atom
- 2015: Visual Studio Code

EVOLUTION OF THE IDE



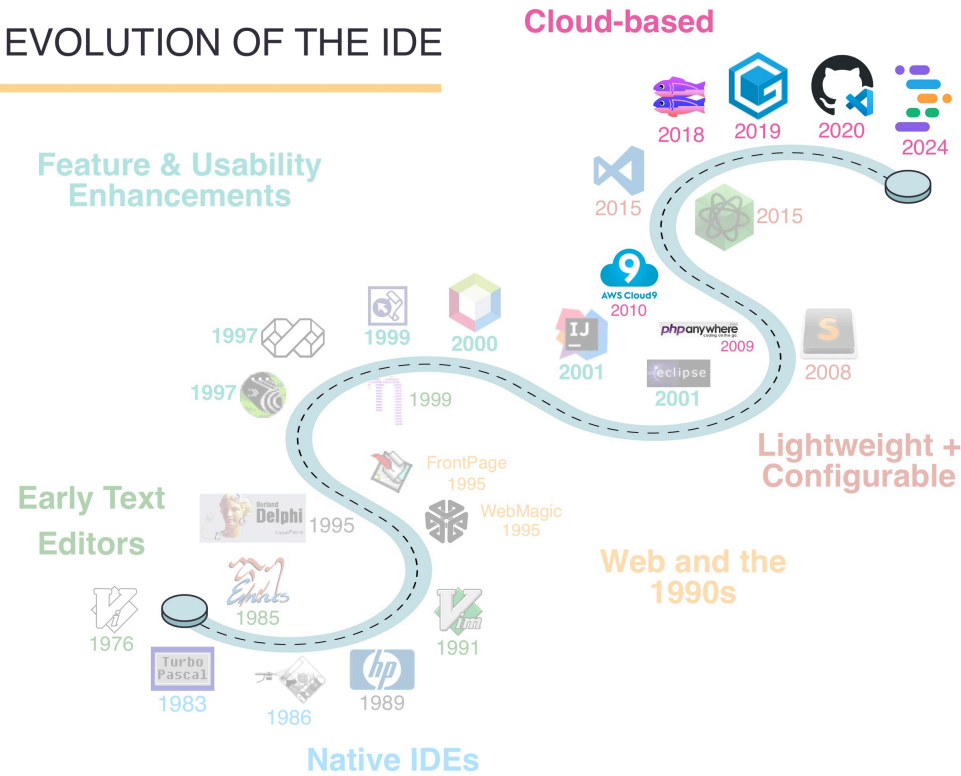
Evolution of the IDE

Cloud-based Options

Now

- 2009: PHPanywhere (eventually becoming CodeAnywhere)
- 2010: Cloud9 (AWS bought it in 2016)
- 2018: Glitch
- 2019: GitPod
- 2020: GitHub Codespaces
- 2024: Google Project IDX

EVOLUTION OF THE IDE



Evolution of the IDE

A result of DevEx

Things we never knew we needed...

From this:

"...the use of an IDE was not well received by developers since it would fence in their creativity."

Evolution of the IDE

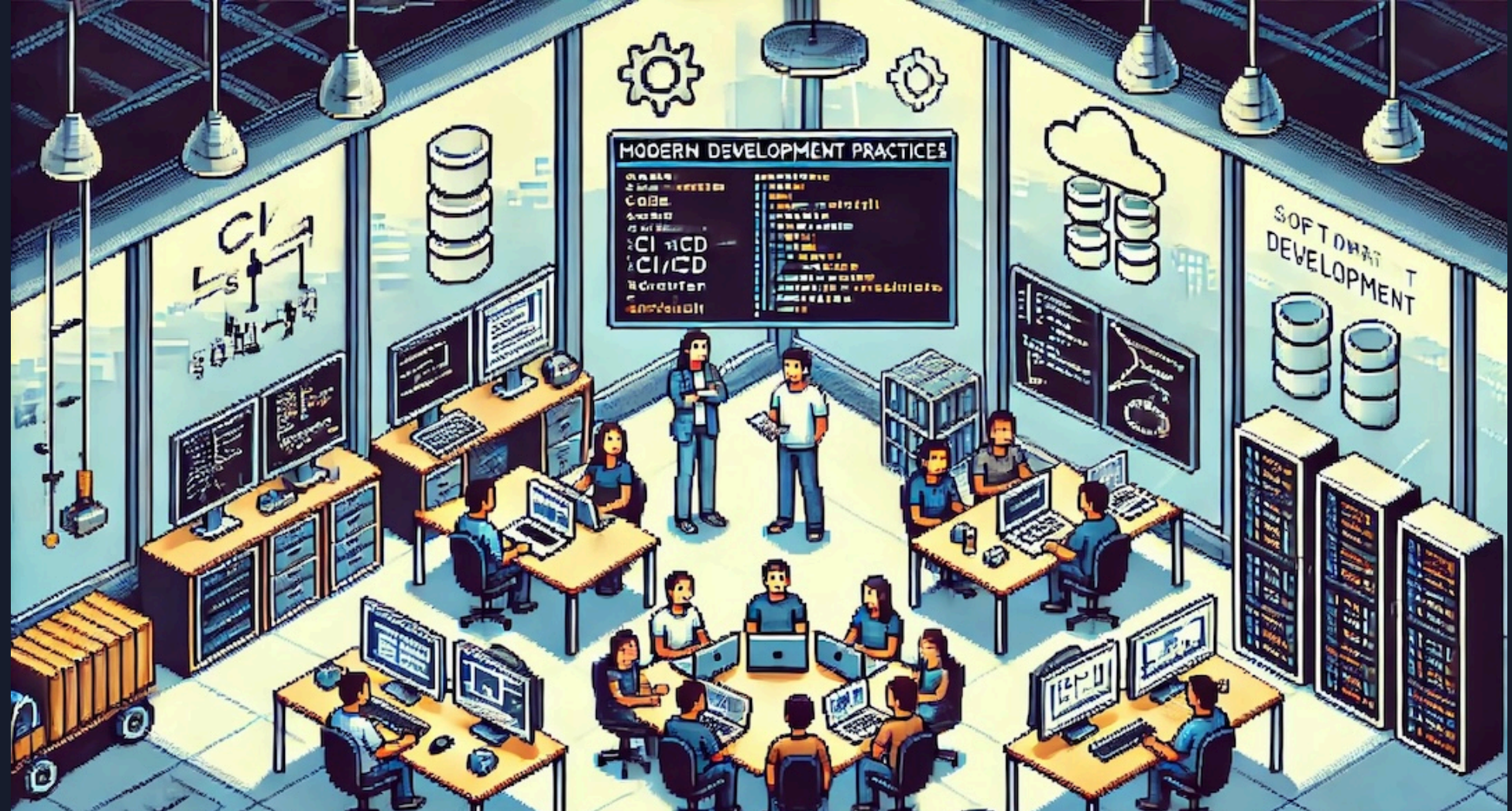
A result of DevEx

Things we never knew we needed...

To this:

- Code completion
- Code refactoring
- Syntax highlighting
- Debugging
- VCS integration (no more FTPing files around)
- Multi-language support
- Framework integration
- Pair programming



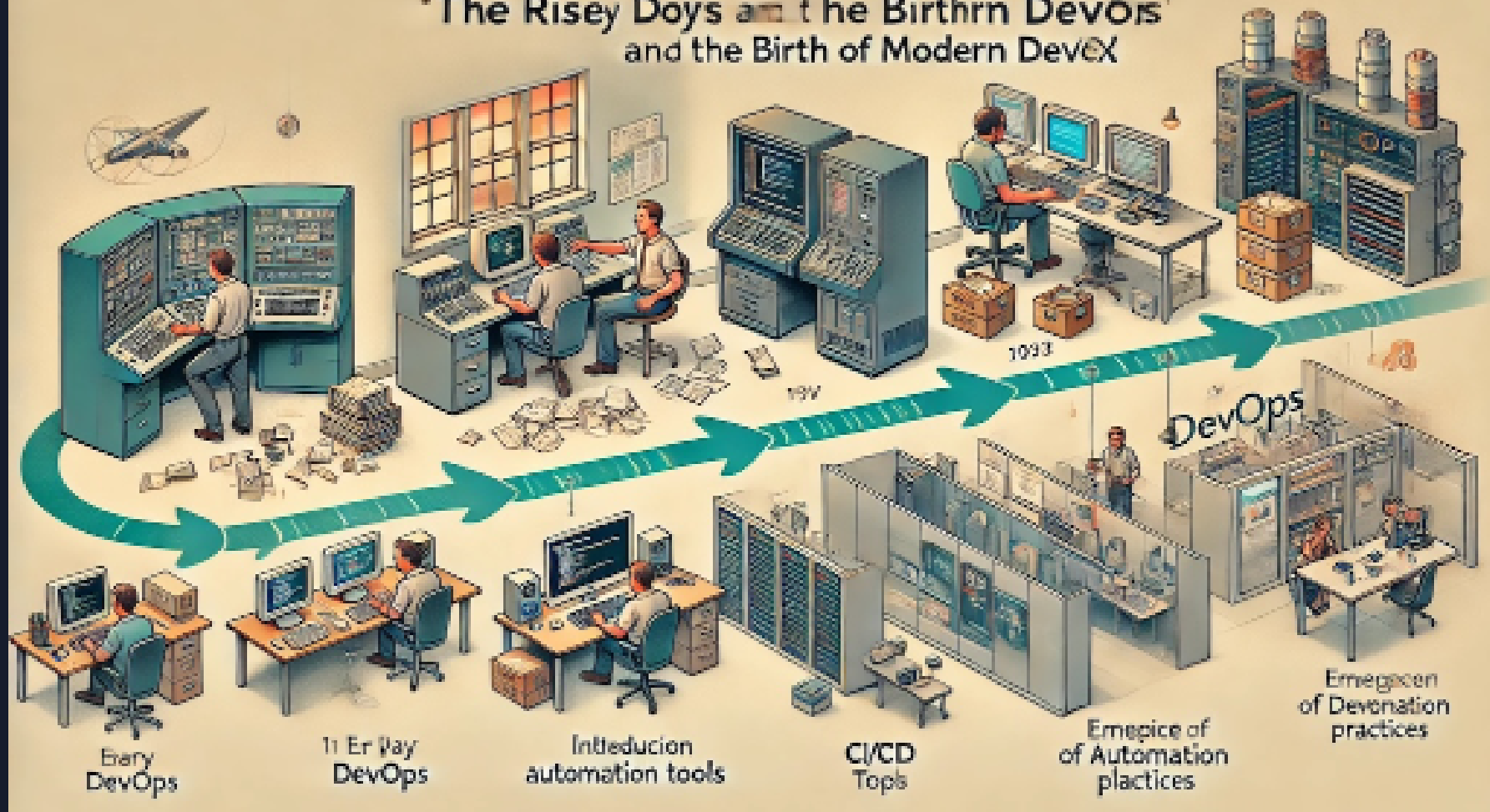


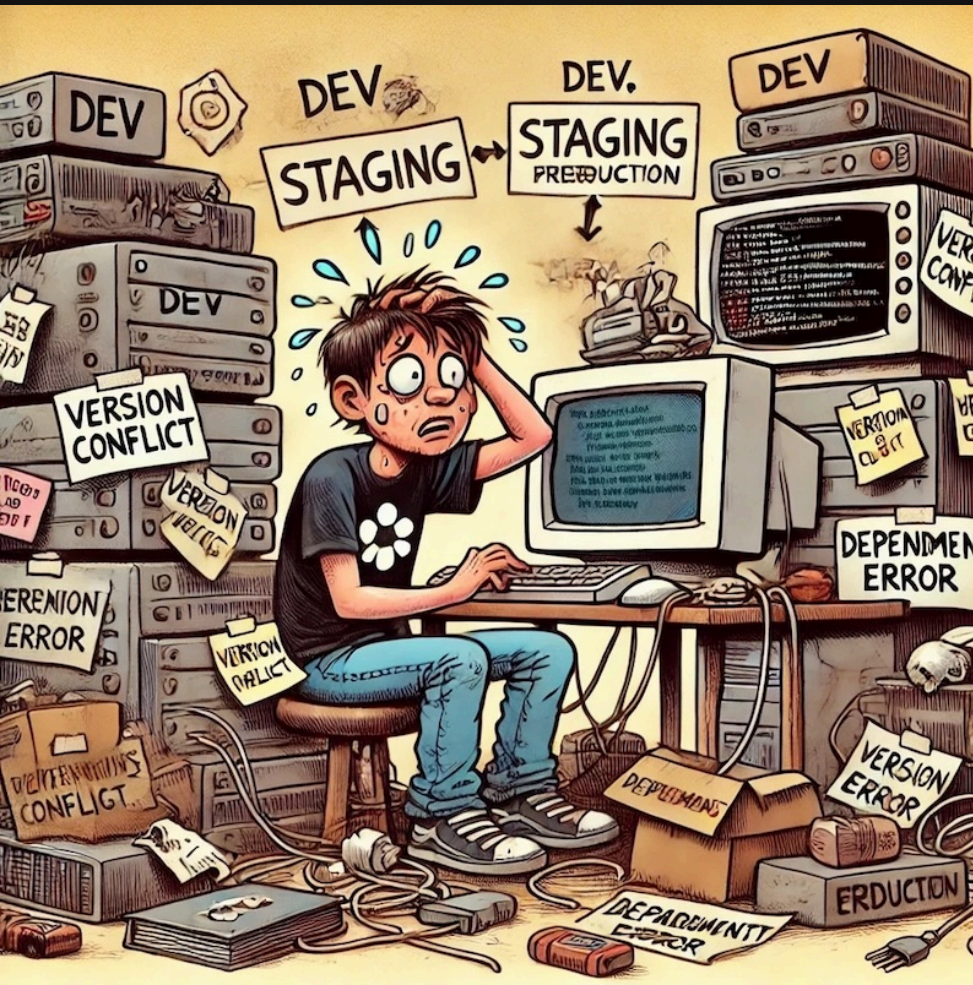
MODERN DEVELOPMENT PRACTICES

- CI/CD
- DevOps
- Agile
- Microservices
- Cloud Native
- Containerization
- Infrastructure as Code
- GitOps
- Observability
- Security as Code
- Automated Testing
- Continuous Deployment
- Feature Flags
- Blue/Green Deployments
- Canary Releases
- Rollback Strategies
- Incident Response
- Post-Mortems
- Blameless Culture
- Documentation
- Knowledge Sharing
- Regular Communication
- Empowerment
- Autonomy
- Transparency
- Collaboration
- Flexibility
- Adaptability
- Resilience
- Scalability
- Performance
- Reliability
- Availability
- Security
- Compliance
- Accessibility
- Localization
- Internationalization
- Performance Optimization
- Load Balancing
- CDN
- Cache
- Database Optimization
- Indexing
- Query Optimization
- Replication
- Sharding
- Partitioning
- Compression
- Encryption
- Backup and Restore
- Disaster Recovery
- Business Continuity
- Incident Response
- Post-Mortems
- Blameless Culture
- Documentation
- Knowledge Sharing
- Regular Communication
- Empowerment
- Autonomy
- Transparency
- Collaboration
- Flexibility
- Adaptability
- Resilience
- Scalability
- Performance
- Reliability
- Availability
- Security
- Compliance
- Accessibility
- Localization
- Internationalization

SOFTWARE DEVELOPMENT

'The Rise of DevOps and the Birth of Modern DevOps' and the Birth of Modern DevOps





Server Environment Setup

Manual config nightmares

Late 1990s to Early 2000s



Server Environment Setup

Config Mgmt & Containerization

Mid-2000s to 2010s



Server Environment Setup

IaC and DevOps Integration

2010s to Present



Broader Impact of DevEx

- Deployment pipeliness
- Infrastructure as Code (IaC) practices
- Developer Efficiencies

What is DevOps?

the combination of practices and tools designed to increase an organization's ability to deliver applications and services faster than traditional software development processes

The Rise of DevOps...

Software Development before DevOps

"It used to take weeks or even months to deploy a simple change."

- Siloed teams with minimal collaboration
- Manual, error-prone deployments
- Lengthy software development cycles

The Emergence of DevOps

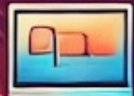
- Collaboration
- Automation
- Continuous Integration

The Role of Automation in DevOps

- Jenkins
- Docker
- Kubernetes

DEVOPS

DEVEX



DEVOPS



AUTOVOPS



AUTOFILCOM



TATHS



ENFUICIENT



SATIFIATION TEAM

AFFUDEFION

DEVEX

CONSIINATION TEAM

DevOps paved the way for Modern DevEx

- Efficiency gains through reduced friction of deployments
- Reduced cognitive load and shift to developer well-being and satisfaction
- Encouragement of experimentation and fast feedback loops



The Birth of Modern DevEx

Cultural Shift towards Collaboration and Experimentation

- Cross-functional teams
- Encouragement of feedback and continuous learning

Core pillars of Developer Experience

Core pillars of Developer Experience

Developer Onboarding

Effective Strategies

- Comprehensive onboarding kits
- Mentorship programs

Measurements

- Time to first commit
- Time to first merge
- " _____ "

Core pillars of Developer Experience

Documentation

- Living documentation
- Developer portals



stripe

Core pillars of Developer Experience

Continuous Feedback

- Regular surveys
- Feedback forums
- Act on feedback

Core pillars of Developer Experience

CI/CD and Automation

- Automate everything possible
- Fast feedback loops ("Fail Fast")



Overcomplicated pipelines can lead to more problems than they solve



Core pillars of Developer Experience

Infrastructure Orchestration

- Developer Self-Service



- Simplifying Deployment



Ensure the tools are well-documented and abstract away unnecessary complexity



Core pillars of Developer Experience

Culture and Team Structure

- Cross-functional teams
- Promote psychological safety

Core pillars of Developer Experience

Developer Well-Being

- Flexible schedules
- Work-life boundaries



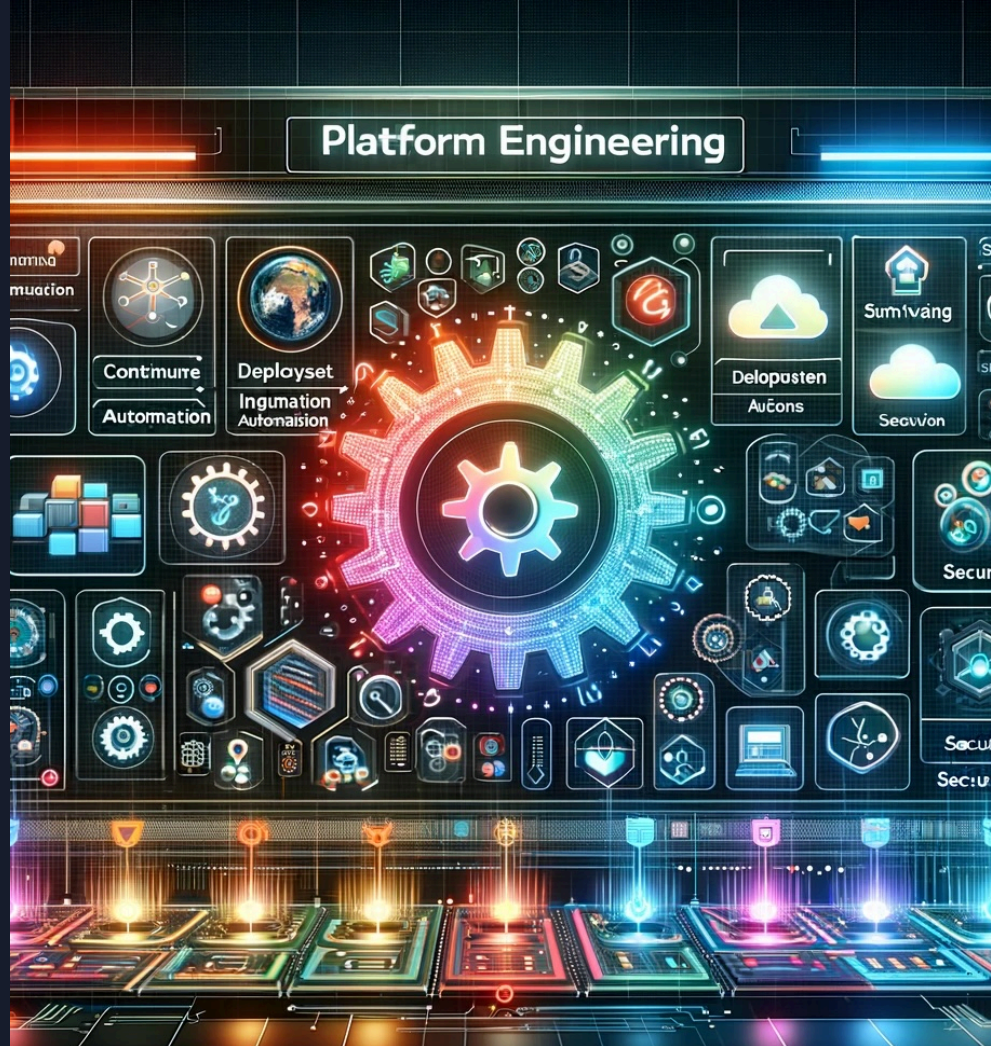
Initiatives like wellness programs, no-meeting days, and social activities can help.





The Rise of Platform Engineering

- Specific, integrated environments that devs need
- Abstract away infrastructure + backend complexities
- Access to robust, scalable, easy-to-use platforms
- Streamline development processes and reduced setup time



Self-Service Platforms

- Developers empowered with necessary tools
- Leverage automation, templates, policies with agility
- Accelerate development, enhance productivity, foster autonomy



Enhancing the Developer Experience

DevEx reflects an organizational culture



Jeremy Meiss, Esq.

@jerdog.dev

If your company does not already have a process for gathering feedback (internal & external) on your product and/or the tools you use, you will not have a good Developer Experience ([#DevEx](#)), and I seriously question the commitment to it.

November 18, 2024 at 4:38 PM  Everybody can reply [↗](#)

Strategies for Improving DevEx

Strategies for Improving DevEx

Improving DevEx in your organization

DevEx initiatives should be modeled from Leadership *FIRST*

Improving DevEx in your organization

1. Foster a positive culture
2. Streamline the workflow(s)



Improving DevEx

Foster a positive culture

1. Clear and concise documentation
 - Encourage knowledge sharing
 - Create easily accessible resources to reduce toil + empower



Improving DevEx

Foster a positive culture

1. Clear and concise documentation
2. Promote collaboration and communication
 - Facilitate code reviews
 - Implement comms to foster teamwork + problem solving



Improving DevEx

Foster a positive culture

1. Clear and concise documentation
2. Promote collaboration and communication
3. Champion well-being and growth
 - Encourage feedback, up and down
 - Recognize achievements
 - Create a sense of belonging



Improving DevEx

Streamline the workflow

1. Tools and Automation

- Explore tools which are highly regarded in your field
- Automate repetitive tasks wherever possible



Improving DevEx

Streamline the workflow

1. Tools and Automation
2. Standardize environment setup
 - Use config management tools
 - Streamline onboarding for all team members

Examples:



Better Practices for leveling up DevEx

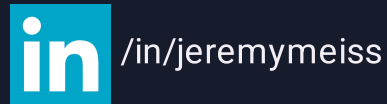


- Empower with the right tools
- Encourage Cross-functional Teams
- Implement Feedback Loops
- Focus on Automation ([Paige Bailey automation post](#))
- Invest in Training and Development

DevEx is...

"ruthlessly eliminating barriers (and blockers) that keep your practitioners from being successful"

Thank you!



END

Good DevOps == Good DevEx

- Facilitates smoother transitions between Dev and Ops
- Minimizes bottlenecks with enhanced collaboration
- Ensures feedback loops are efficient and productive
- Enables DevOps principles to take hold within an organization