Affective and Conative Patterns for Improving Developer Experience

DANIEL PINHO and ADEMAR AGUIAR, Faculty of Engineering, University of Porto and INESC TEC

VASCO AMARAL, NOVA LINCS, DI, FCT/UNL

In a world where software systems have become highly pervasive and a part of our daily lives, we have seen a growing need for software developers. These professionals have their own needs due to the mental nature of their work. Based on user experience, the concept of developer experience (DX) was defined with a focus on the dual role of user and creator of software that developers have in their work. This paper continues our work on a pattern language that aims to assist developers with improving their DX, having previously explored the cognitive dimension of DX, related to developers' perceptions of their infrastructure. In this work, we present patterns related to the affective dimension of DX, which deals with how developers feel about their work: Good Feelings, Stakeholder Relationships, and Team Players. We also dive into the conative dimension, related to how developers see the value of their contributions, with the patterns Well-Valued Contribution and Intrinsic Motivation.

CCS Concepts: **Software and its engineering** → **Software creation and management** → **Software development process management** → **Software development methods** → **Design patterns** • Human-centered computing → Human computer interaction (HCI) → HCI theory, concepts and models

Additional Key Words and Phrases: developer experience, patterns

ACM Reference Format:

Pinho, D. and Aguiar, A. and Amaral, V. 2024. Affective and Conative Patterns for Improving Developer Experience. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 31 (October 2024), 15 pages.

1. INTRODUCTION

Over the past decades, software has increasingly become a staple in our daily lives. From systems that require precision or added care, such as those found in financial and medical fields — to those that we use for our entertainment, such as video streaming services, video games, and e-books — we are sure to interact with a plurality of software systems every day.

This prominence and prevalence of software solutions does not appear by itself; each system has been built by someone who can go from a singular developer working on a pet project in their spare time to a large enterprise where we find development teams, coaches, managers, and stakeholders, along with other roles, working on a large-scale enterprise product. At its core, software development activities require developers to implement new

This work was funded by the FCT - Fundação para a Ciência e Tecnologia grant 2021.08371.BD.
Corresponding author's address: Daniel Pinho, Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal; email: daniel.pinho [at] fe.up.pt;

features and maintain existing ones. With this, coupled with the high demand for developers and their skills [Breaux and Moritz 2021], we are incentivised to look into their needs.

Building upon the idea of user experience (UX), which takes into account the perceptions and responses that come from a person using a given system, Fagerholm and Münch [Fagerholm and Münch 2012] defined the concept of developer experience (DX). While UX focuses on what goes on as a person *uses* a tool, DX redirects its attention to the dual role of *creator* and *user* that a developer has; as software development is a mental task, developers are affected by the tools they are using (e.g. through usability concerns), while also having to deal with the mental load that comes with interacting with other people and building something that will have its own users at the same time. This added responsibility brings some differences in the needs of developers, especially when compared to users.

This work expands upon previous articles where we documented a set of pattern candidates for a potential pattern language focusing on improving the developer experience. The first article [Pinho et al. 2023] focused on low-code contexts, while the second [Pinho et al. 2024] targeted developers at large but focused on the cognitive dimension of DX, i.e. how developers perceive their development infrastructure. This paper targets the other two dimensions of DX, affective and conative, by introducing more pattern candidates. The affective dimension approaches the feelings a developer has towards their work, while the conative dimension relates to how developers feel about their contributions.

The remainder of this paper is structured as follows: Section 2 provides background on UX and DX. Section 3 provides an overview of the tentative pattern language. Section 4 presents the GOOD FEELINGS pattern candidate, which touches directly upon the affective dimension of the DX framework based on Fagerholm and Münch's definition. Section 5 showcases WELL-VALUED CONTRIBUTION which, similarly to the previous pattern, also touches on a dimension of DX, albeit the conative one in this case. Section 6 deals with the relationships between developers and other stakeholders, a part of the affective dimension. Section 7 continues with the focus on the affective dimension with TEAM PLAYERS, which focuses on teamwork. Section 8 shifts gears into the conative dimension with INTRINSIC MOTIVATION, discussing the factors that motivate a developer. Finally, we can find in Section 9 concluding remarks to this work.

2. BACKGROUND

This section includes some background information about the main topic approached by the pattern language, developer experience, after delving into the basics of user experience.

2.1 User experience

User Experience (UX) is defined by the ISO 9241-11 standard as being the set of a "user's perceptions and responses that result from the use and/or anticipated use of a system, product or service" [International Organization for Standardization 2018]. These perceptions and responses include but are not limited to feelings, beliefs, behaviours and perceptions. Additionally, the standard also mentions that UX is a consequence of several factors, such as functionality, brand image, and system performance. The user's personal context also influences the UX, and their prior experiences, abilities, personality and outlook should be taken into account [International Organization for Standardization 2018]. This definition is supported by authors such as Law et al. [Law et al. 2009], who state that UX is a subjective concept that is dynamic and dependent on context, and McNamara and Kirakowski [McNamara and Kirakowski 2006], who argue that UX deals with the relationship between the user and the product.

Usability is another concept with a definition in the ISO 9241-11 standard, stating that it is the ability for a product to be used to achieve "specified goals with effectiveness, efficiency and satisfaction" [International Organization for Standardization 2018] in a given context. The Nielsen-Norman group emphasize the difference between usability and UX: the former focuses on ease of use [Nielsen 2012], while the latter is a broader concept [Norman and Nielsen 2021].

2.2 Developer experience

Developer Experience (DX) was coined by Fagerholm and Münch [Fagerholm and Münch 2012] as a response to the fact that developers have a dual role of user-creator, leading to the definition of UX not covering all of their needs. Software development is a highly mental task, encompassing both hard and soft skills, which leads to developers having varied goals and tasks [Cockburn 2006]; for example, in a work day, a programmer can be in meetings with clients, engage in brainstorming sessions with other developers, and work on implementing features or fixing bug, all with the overarching goal of building a system that others will use. This brings additional factors to consider regarding what a developer experiences compared to regular end-users. The DX concept targets everyone that is involved in software development activities.

Fagerholm and Münch included different aspects based on the trilogy of mind theory commonly discussed in psychology [Hilgard 1980]. These aspects are the cognitive dimension, related to developers' perception of the infrastructure they interact with; the affective dimension, which deals with how developers feel about their work; and the conative dimension, linked to the value developers give their work [Fagerholm and Münch 2012].

3. PATTERNS FOR DEVELOPER EXPERIENCE

This section describes the process used for pattern mining, the pattern language we're presenting, and sets the stage for the pattern candidates discussed in this paper.

3.1 Pattern mining process

In the process of documenting patterns for this pattern language, we used a pattern mining process that considered two different sources of information: *academic publications* and *industry-based knowledge*.

For academic publications, we considered peer-reviewed works that discussed DX or touched upon a topic mentioned in the framework as discussed in Fagerholm and Münch's work [Fagerholm and Münch 2012]. We used academic search engines such as Google Scholar[1], Semantic Scholar[2], and Scopus[3].

Regarding the second realm, industry knowledge, two types of information were gathered: first, based on previous experience on the authors' side, including work on software projects and in our experience in software engineering education. Secondly, we also looked at information that is available in the grey literature, such as development tools' websites and blog posts.

3.2 Pattern language overview

The patterns described here use a structure with explicit sections, and are composed of name, context, problem, forces, solution, and consequences. Relationships between patterns, whether they are from this pattern language or another, are described whenever relevant in their own heading. The pattern language primarily targets developers as its audience, but recognises that there are factors that, while important to a developer's DX, are outside of their sphere of influence. As such, we see that there is value in stakeholders (such as project managers and clients) as well as development tool makers being aware of what is discussed in this pattern language.

The pattern language considers three main types of patterns: *Cognitive*, *Affective*, and *Conative*. Each one of these maps themselves to the mental dimensions of the same names present in Fagerholm and Münch's definition of DX [Fagerholm and Münch 2012]. A pattern is not restricted to having one type; depending on the relationships between the patterns leading to it (see Figure 1), their influence may extend beyond just one of the mental dimensions of DX. As starting points, the pattern candidates INFRASTRUCTURE ALIGNMENT, GOOD FEELINGS, and WELL-VALUED CONTRIBUTION are direct mappings to each type of pattern: cognitive, affective, and conative, respectively.

[1] `https://scholar.google.com/`
[2] `https://semanticscholar.org/`
[3] `https://www.scopus.com/`

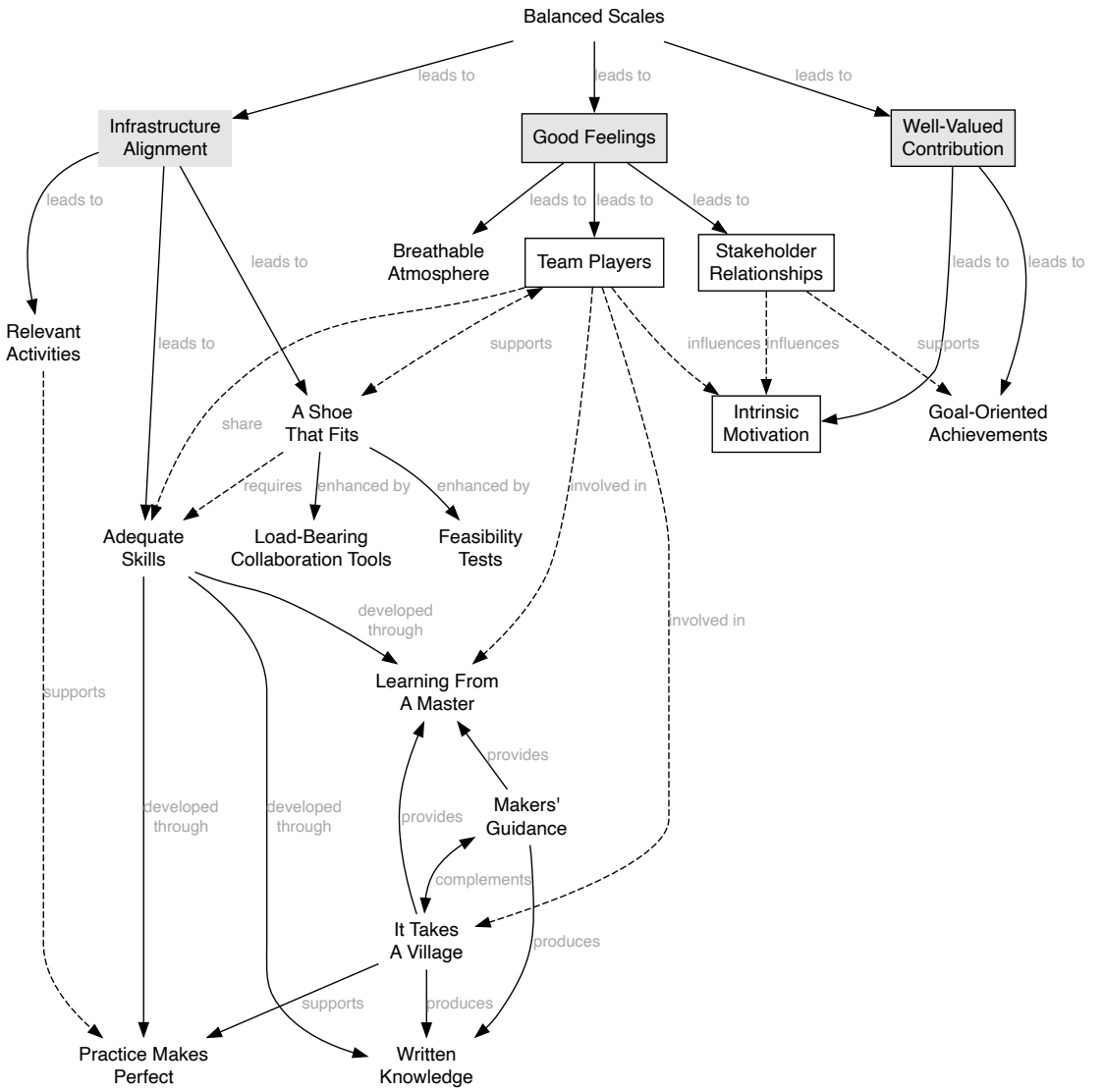


Fig. 1. The candidate patterns for the pattern language. A black rectangle outline indicates the patterns that are described in their entirety, while pattern candidates directly mapped to a DX dimension are shaded in gray.

Figure 1 provides an overview of the pattern candidates and how they relate to each other. Patterns described in their entirety are signalled with a rectangle outline, while the three patterns that are directly mapped to each of the DX dimensions are shaded in gray. Full arrows indicate that the usage of a pattern leads to another one, akin to a pattern sequence; meanwhile, dashed arrows indicate patterns that influence other ones.

Table I provides details on all pattern candidates in the pattern language, enabling the reader to be contextualised regarding their contents, particularly when these are mentioned in the *Related Patterns* section of patterns. The pattern candidates discussed at length in this paper are highlighted in bold. Pattern candidates first discussed in

Table I. Patlets of the pattern candidates.

Pattern	Type	Patlet
Balanced Scales [Pinho et al. 2023]	Cognitive, Affective, Conative	**P:** How can we maximise the quality of DX-influencing factors? · **S:** Assess the impact of the different factors around have on each one of the dimensions of the mind: cognitive, affective, and conative. Considering any limitations in place, assess the importance of each dimension relative to the other ones and focus more on improving the most important ones.
Infrastructure Alignment [Pinho et al. 2024]	Cognitive	**P:** How can we maximise the effectiveness of the development tool in use? · **S:** Reflect on the conditions that influence your performance with the tasks that need to be done, improving upon them if possible. Develop your skills, assess whether the tool is right for the job, and review the activities performed for their usefulness.
Good Feelings	Affective	**P:** How can we strive for good feelings about our work? · **S:** Developers should strive for a positive social environment with teamwork, good relationships with stakeholders, respect, trust, and belonging to bolster their feelings.
Well-Valued Contribution	Conative	**P:** How can developers see the value of their own contributions? · **S:** Assess factors related to one's own value that are compatible with the project, including motivation, goals, and interpersonal alignment.
Goal-Oriented Achievements	Conative	**P:** How should goals be defined? · **S:** Adopt strategies for effective goal-setting, such as bringing developers and stakeholders together in the goal-making decision, employing a mix of large-scope and smaller-scope goals, defining SMART goals, and employing mechanisms to track their progress.
Intrinsic Motivation	Conative	**P:** How can developers boost their motivation? · **S:** Developers ought to reflect and assess their working context — their environment, career, tasks, and the people around them — to identify what they have that motivates them and what can be improved.
Stakeholder Relationships	Affective	**P:** How can we reduce friction between developers and other stakeholders? · **S:** Consider fostering a harmonious environment where effective communication strategies are employed and building trust to strengthen stakeholder relationships.
A Shoe That Fits	Cognitive	**P:** What kind of tools should developers opt for? · **S:** Developers should use tools that not only enable the work to be done but also support it, taking into account factors including usability, skill, and ubiquity, and interoperability.
Feasibility Tests [Pinho et al. 2023]	Cognitive	**P:** How can tool makers be sure that their tools have good usability? · **S:** At regular time intervals, usability-oriented tests should be conducted on the tools, such as the NASA TLX or the System Usability Scale, to prevent usability issues in the long run.
Team Players	Affective	**P:** What should a set of developers do to become a team? · **S:** Have developers come together with an aligned approach, working together, sharing knowledge, supporting each other, and applying a process that works for them, while being mindful of each other's personalities.
Adequate Skills [Pinho et al. 2024]	Cognitive	**P:** How can we best learn how to use a development tool? · **S:** Engage in a multifaceted skill-building process that includes practice, written materials, guidance from experts, and other available resources.
Written Knowledge [Pinho et al. 2024]	Cognitive	**P:** What approaches should developers take to learn more about the tools and processes in use? · **S:** Developers should consider expanding their theoretical base knowledge through written sources of information, such as books, documentation, guides, and tutorials.
Practice Makes Perfect [Pinho et al. 2024]	Cognitive	**P:** How can developers effectively develop their skills with the tools and processes they use? · **S:** Consider acquiring first-hand experience through practice exercises and regular daily work to learn more about the intricacies of the tools and processes in use.
Learning From A Master [Pinho et al. 2024]	Cognitive	**P:** How can developers learn effectively to maximise their proficiency with their tools and processes? · **S:** Developers should seek guidance and mentorship from other more experienced developers or instructors via classes, workshops, or simply by asking questions.
It Takes A Village [Pinho et al. 2024]	Cognitive	**P:** How can we find pertinent knowledge to develop our skills? · **S:** Consider joining and being part of the community associated with the tools in use, interacting and sharing knowledge with other developers.
Makers' Guidance [Pinho et al. 2023]	Cognitive	**P:** How can developers acquire knowledge on how to use their tools effectively? · **S:** Seek out informative materials coming directly from the makers of the tools, such as documentation, guides, tutorials, and classes, while building bridges with them.
Breathable Atmosphere	Affective	**P:** What factors into a positive work environment? · **S:** Organisations should foster the psychological safety of their developers, giving them the space to speak up, share ideas, take initiative, and build working relationships with their coworkers.
Relevant Activities	Cognitive	**P:** What activities should we focus our time on? · **S:** Select activities that are aligned with the main objectives of the project, eliminating spurious tasks.

previous work (such as in [Pinho et al. 2023] and [Pinho et al. 2024]) have the citations next to their names in the table.

4. PATTERN: GOOD FEELINGS

Context

Software development is a highly mental endeavour, relying on our human abilities to build systems [Cockburn 2006].

Problem

Humans are social and emotional creatures; their happiness, comfort and other emotions affect how they perform, with positive feelings having a beneficial effect [Graziotin et al. 2018; Kurian and Thomas 2023].

How can one foster and maintain good head-space during their work time?

Forces

Is our teamwork making the dream work?. Developers do not often work alone, as most software projects are collaborative efforts. The composition of a person's team, the relationships between its members, and the personalities within the team can strongly influence their experience.

Are the social interactions productive?. In software development environments, people with varied roles interact with each other, from developers to business people to clients. These interactions can enrich developers' productivity with different perspectives being brought in, but the differences in backgrounds, responsibilities and expectations can also bring interpersonal clashes.

How is the environment?. An atmosphere that welcomes developers and other professionals should make them feel more at ease, but different people will react to different factors.

Solution

Build strong, positive, trusting working relationships with teammates, work on building bridges with stakeholders, and contribute towards a positive atmosphere where there is psychological safety and one's needs can be met.

A developer's teammates are most likely the people they interact with most during a workday. These interactions range from collaborating on tasks, coordinating work in meetings, and sharing knowledge and experience. A developer's team should be a well-oiled machine and provide an atmosphere where trust is prevalent, team

members can rely on each other, and everyone can have their own space. The pattern TEAM PLAYERS dives further into this topic.

Depending on the size of their organisation, a developer may encounter people with other roles besides their teammates, such as business people, agile coaches, clients or end-users, and even other developers. We can find people with different backgrounds and experiences in these varied roles, which suggests that people should be open and communicative to ensure smooth cooperation between the various stakeholders. The pattern STAKEHOLDER RELATIONSHIPS examines this topic in further detail.

Even if developers and their coworkers interact well enough to get things done, the right social atmosphere is vital to further boost developers' mindsets and productivity. This atmosphere should be built on trust, respect, and belonging. The pattern BREATHABLE ATMOSPHERE delves further into this.

Consequences

Nurturing relationships and a positive work atmosphere have beneficial results, as developers perform better when they are happy.

However, depending on the situation, reaching these states may be easier said than done, as human interaction goes; some people may find it hard to get along with each other or their needs may not mesh well with their colleagues, and achieving a state of high spirits may require a good investment of time and money.

Related Patterns

The solution of this pattern points to TEAM PLAYERS, STAKEHOLDER RELATIONSHIPS, and BREATHABLE ATMOSPHERE. Compare and contrast this pattern with the other two that deal directly with the dimensions of DX: INFRASTRUCTURE ALIGNMENT and WELL-VALUED CONTRIBUTION.

5. PATTERN: WELL-VALUED CONTRIBUTION

Context

Like all people, software developers have their own views and values, which can influence how they act.

Problem

In software development contexts, the actions and decisions made by the developers and other personnel happen with the projects' completion as the ultimate goal. This can conflict with a developer's world-view and set of beliefs, especially if the project is large or if there is a disconnect between the project's scope and the developer's values.

How can developers see the value of their own contributions?

Forces

What motivates us?. Motivation can be a strong driver for one's work. Finding what keeps us motivated can increase our productivity.

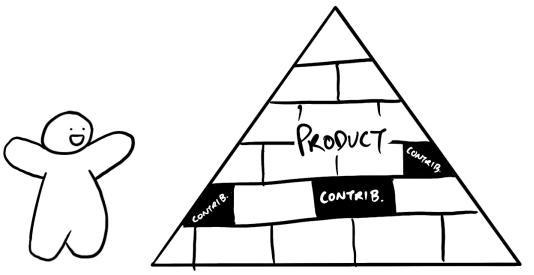
What are our goals?. Having well-defined goals can bring us clarity, assisting us in identifying the next steps we need to take.

Is everyone on the same page?. Ensuring everybody is aligned with each other reduces repeated or irrelevant work and ensures work is being done towards the same objective.

Solution

Assess factors related to one's own value that are compatible with the project, including motivation, goals, and interpersonal alignment.

According to Deci and Ryan [Deci and Ryan 2012], partaking in a task as a means to an end is a manifestation of extrinsic motivation; for instance, a developer may work at their current job because of the pay they receive at



the end of the month, even if they do not enjoy other facets of their work. On the other hand, intrinsic motivation stems from the task itself, with it being pleasurable and satisfying [Kuusinen 2016]. Reflecting on work tasks and identifying which ones provide intrinsic motivation can give a developer a new perspective on their work. The pattern INTRINSIC MOTIVATION examines this idea in further detail.

Having clear-cut and well-built goals is helpful in that developers know what they should do and why they are doing it, and it reduces the potential for them to feel lost regarding their work [Locke et al. 1981]. GOAL-ORIENTED ACHIEVEMENTS provides strategies for effective goal-setting.

Developers usually work with other people, so they should work towards having a greater awareness of everyone's contribution. This alignment process enables developers to know how all tasks fit into the bigger picture while providing an opportunity for sharing feedback.

Consequences

Exploring these factors makes it possible for developers to learn how valuable they are to the project they are working on, improving their DX by enhancing their sense of self-worth. However, this reflection and its related methods require time and care to be well-executed.

Related Patterns

The solution of this pattern points to INTRINSIC MOTIVATION and GOAL-ORIENTED ACHIEVEMENTS. Compare and contrast this pattern with the other two that deal directly with the dimensions of DX: INFRASTRUCTURE ALIGNMENT and GOOD FEELINGS ABOUT WORK.

6. PATTERN: STAKEHOLDER RELATIONSHIPS

Context

Software projects are collaborative efforts, with people from various backgrounds and performing myriad roles working together to reach a common goal. Developers, alongside project managers, users, and product owners, are examples of stakeholders: people who are invested in the project and want to see it through.

Problem

The inherent differences in viewpoints that come from having differing knowledge, specialisations, and tasks can be a hurdle to maintaining positive relationships between developers and other stakeholders, which can, in turn, hinder the developers' experience.

How can we reduce friction between developers and other stakeholders?

Forces

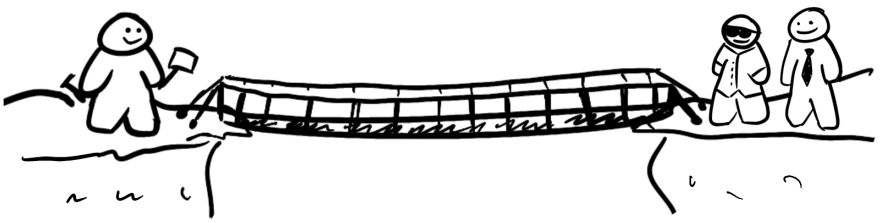
How many cooks are there in the kitchen?. The number of people actively working on a project can influence the decisions that are made, with each stakeholder having a view on goals, priorities, and values.

How present are we?. While developers may be solely dedicated to one project, other stakeholders, like customers and organisation managers, may split their attention between multiple projects.

What do we know?. A stakeholder's world-view will be influenced by their knowledge and skills, giving them differing perspectives on topics such as technical details, jargon, and specificities on business rules.

Solution

Work towards a harmonious environment where effective communication strategies are employed and building trust to strengthen stakeholder relationships.



The literature has established that effective communication is a critical component in the relationship between developers and other stakeholders [Galli 2019; Kuusinen 2018; Poole 2003]. As all stakeholders want the project to succeed [Poole 2003], they should strive towards minimising potential liabilities and weak points in their processes.

Each type of stakeholder has their own motivators, goals, and responsibilities that they need to attend to. For instance, while a developer is focused on programming and other development tasks and can be driven by wanting to learn new technologies, a project manager can work on keeping the team on schedule and budget while meeting with other stakeholders to assess their needs [Poole 2003]. To avoid potential disagreements on priorities, technical opinions, resources and scheduling, the stakeholders should identify each role and its needs to ensure they understand where their colleagues are coming from and where their points of view lie [Galli 2019; Kuusinen 2018].

Effective communication techniques include frequent and transparent communication, continuously updating the project calendar [Poole 2003], understanding shared goals, maintaining a receptive attitude, performing active listening, and employing critical thinking [Burnett 2005].

It is worthwhile for stakeholders to develop trust, as it reduces communication barriers. For instance, if something happens during a software project, a developer may be more confident in having a potentially uncomfortable conversation with a client or another stakeholder if they know they have each other's trust in seeing the project through. Trust can be built by encouraging participation, providing feedback, and practising accountability by admitting mistakes [Galli 2019; Lencioni 2006].

Support materials, such as comprehensive documentation, glossaries, and value maps, can be an additional tool in the stakeholders' belt to ensure everyone is informed and up to date on current project developments [Galli 2019; Kuusinen 2018], which is crucial in maintaining a high-quality level of communication. These are worth having when project personnel changes, as tacit knowledge is hard to record, and domain-specific jargon can induce misunderstandings due to differences in background [Galli 2019].

Consequences

Fostering positive relationships between developers and other stakeholders can assist in achieving a smoother developer experience by minimising conflict and promoting a good social environment.

On the other hand, developers cannot employ this pattern by themselves, needing support from the rest of the stakeholders to implement these practices. It is also worth noting that developers can still be unhappy if they see they are not a good fit for the organisational culture. The pattern BREATHABLE ATMOSPHERE aims to target this last point.

Related Patterns

The pattern TEAM PLAYERS shifts the scope and goes into details about the relationships that exist within the team. As mentioned in this pattern's consequences, BREATHABLE ATMOSPHERE deals with organisation culture, which can also influence how relationships between stakeholders play out.

The stakeholders can have a significant role in how goals are defined; see GOAL-ORIENTED ACHIEVEMENTS for more on that.

7. PATTERN: TEAM PLAYERS

Context

Software projects are often collaborative efforts, with multiple people of varying backgrounds, roles and skill sets working together.

Problem

It can be intuitively inferred that teams are made of people working to reach a common goal, but there are intricacies in the difference between a group of people and a team.

What should a set of developers do to become a team?

Forces

What do we have our sights on?. A team's direction, which depends on their goals and tasks, influences their results.

Who is manning the helm?. The leadership quality (whether focused on one person or distributed) influences the team's ability to get things done. Some teams have one person in a leadership role, such as a team leader who is a developer, while others are self-organised, including everyone in the decision process.

Who do we have on our team?. A team is composed of people working towards the same goal. The interactions and relationships between them, along with their skill sets, personalities, and preferences, influence how work will be carried out.

How are things organised?. The organisation in and outside the team influences its performance. The processes in place and management affect the team's results.

Solution

Have developers come together with an aligned approach, working together, sharing knowledge, supporting each other, and applying a process that works for them, while being mindful of each other's personalities.

A team aligns itself towards a common objective or goal, often the project's completion. This alignment can be guided into other goals that are smaller in scope, as is the case for the Sprint Goal in Scrum teams. The team's leadership can influence their direction. Some teams are self-organised, collectively making decisions, assessing tasks, and setting goals. In contrast, others may have their leadership concentrated on one person, as for team leaders in enterprise contexts. Both team types can suffer through poor leadership; a supervisor or a team leader may lack the necessary skills or a cohesive vision to guide their team members best, while self-organised teams are challenging to implement [Moe et al. 2010].

Developers should consider working with their teammates on the same tasks, such as performing pair programming. It enhances their skill-building and enables the easy spread of tacit knowledge within the team. The role played by relationships is also of note; building relationships can help with productivity [Elizalde and Bayona 2018], and working together is an easy way of doing it through frequent interactions facilitated by pair work [Cockburn 2006].

Working together can additionally pay dividends whenever a team member is away and requires someone to back them up by continuing their work; team members will feel more at ease with such a task if they have prior knowledge (coming from already having worked together on the same parts of the project) and are invested in supporting their colleagues [Cockburn 2006; Moe et al. 2010].

Differences in personalities, particularly along the introvert-extrovert spectrum, need to be acknowledged within the team. Introverts need to have opportunities to withdraw from the collective and recharge their social batteries [Griffith 2016].

Consequences

Positive interpersonal contact helps with productivity, and interactions between team members are a source of knowledge transfer and skill building. Overall, having a good working relationship with their teammates can help a developer feel better about their work and environment.

However, teams are still a complex topic. Soft skills are a requirement to effectively work in teams due to the cooperative and interdisciplinary nature of software development, with people from different backgrounds and world-views working together. In addition, team members need to mesh together; if their personalities and social preferences get in the way, the developers might be better off in different teams.

Related Patterns

The patterns STAKEHOLDER RELATIONSHIPS and BREATHABLE ATMOSPHERE are similar patterns that affect the developer's feelings about their work, focusing on the relationships between developers and stakeholders and the overall workplace culture and psychological safety, respectively.

Good teamwork can affect a developer's motivation; the pattern INTRINSIC MOTIVATION delves further into this. The team can also help with skill building; see the patterns ADEQUATE SKILLS and LEARNING FROM A MASTER for more details.

The concept of teamwork and how team members should interact is also approached in other pattern languages. The patterns COMMUNITY OF TRUST, SACRIFICE ONE PERSON, DEVELOPING IN PAIRS, MANAGER ROLE, and SELF-SELECTING TEAM are examples of patterns that dive deeper into the intricacies of teamwork from Coplien and Harrison's *Organisational Patterns* [Coplien and Harrison 2004]. One can also find examples of teamwork-

related patterns in the *Scrum Patterns* [Sutherland et al. 2019], such as COLOCATED TEAM, SELF-ORGANIZING TEAM, and AUTONOMOUS TEAM.

8. PATTERN: INTRINSIC MOTIVATION

Context

Software development is a highly mental process, where developers must focus and exert a mental effort on the tasks they work on in each step of the software development life-cycle.

Problem

Having the drive to exert this effort can be challenging if developers feel unmotivated, which can result from myriad factors.

How can developers boost their motivation?

Forces

What is in our surroundings?. Our environment influences how we feel. Having good conditions in which we can work can positively influence our motivation.

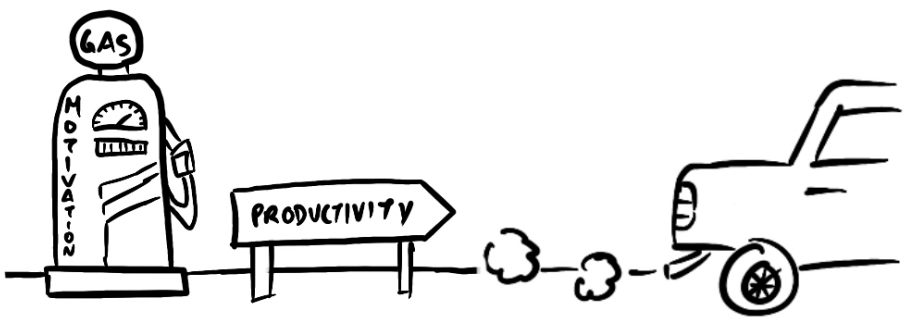
What are we doing?. Our tasks and goals and what they bring to the table (e.g. variety, a challenge, opportunities for growth) will influence how we see our work.

Who do we have around us?. Developers interact with others in their work, such as team members and stakeholders. Working with effective leaders and having good relationships contributes to one's motivation.

Can we change our world?. The scope of one's sphere of influence can dictate what we can do to improve our motivation.

Solution

Developers ought to reflect and assess their working context — their environment, career, tasks, and the people around them — to identify what they have that motivates them and what can be improved, bringing in the aid of stakeholders and organisation personnel whenever needed.



Intrinsic motivation comes from inside a person's mind, in contrast to extrinsic motivation, which stems from external rewards. As such, the former is influenced by varying factors related to one's environment, career, tasks, and social interactions [Deci and Ryan 2012].

For instance, common motivators related to one's tasks and career include identifying with the task at hand and finding it interesting. Seeing career growth options and being challenged are also factors that, if present, boost one's motivation. On the other hand, feeling stressed, lacking career growth opportunities, or feeling inequity are all factors that cause a developer's motivation to decrease [Beecham et al. 2008].

Motivators related to one's physical and social environments include having one's needs met, being under good management, and having positive relationships alongside a sense of belonging. Factors that hinder one's motivation include poor communication, bad relationships, and a lack of resources [Beecham et al. 2008].

Becoming aware of these motivating factors and recognising when they are present can bring a renewed sense of motivation to a developer. However, identifying which de-motivators are present can also bring value if one can act on removing them or reducing their impact. For example, a developer can try to improve their relationships with their teammates or enlist the support of other co-workers to bring change to their organisation.

Consequences

Motivated developers are more productive, which has the added benefit of helping them see their contribution more positively.

However, not all motivation-related factors fall into a developer's sphere of influence and are not directly actionable, even with the aid of other organisation personnel, which can end up being a cause of frustration on its own.

Related Patterns

This pattern is one that contributes directly to WELL-VALUED CONTRIBUTION, alongside GOAL-ORIENTED ACHIEVEMENTS.

The patterns TEAM PLAYERS and STAKEHOLDER RELATIONSHIPS can guide developers into improving their social environment, positively impacting their motivation. The patterns ADEQUATE SKILLS and RELEVANT ACTIVITIES dive deeper into task and skill management.

9. CONCLUDING REMARKS AND FUTURE WORK

The variety of software systems we interact with every day is maintained by a staggering number of developers and development teams worldwide. The demand for developers and their skills drives us to address their needs, as is the case with the concept of developer experience.

In this paper we looked at the literature and the industry to document patterns that positively impact the DX encountered by developers while working on their tasks, continuing the work from previous papers. Alongside a set of pattern candidates, two of the patterns in this paper discussed in more detail, GOOD FEELINGS and WELL-VALUED CONTRIBUTION, give us insight into how we should approach the affective and conative dimensions of a developer's mind. The remaining patterns dive deeper into these two dimensions; STAKEHOLDER RELATIONSHIPS gives advice on how to communicate effectively with other stakeholders to reduce conflict, TEAM PLAYERS discusses the role teamwork has in a team, and INTRINSIC MOTIVATION looks at which factors influence a developer's motivation<.

In the future, we expect to continue work on this pattern language, documenting more patterns, revising existing documented ones and strengthening the relationships between them.

ACKNOWLEDGMENTS

We would like to thank Hironori Washizaki for his feedback during this paper's shepherding process. We also thank Cesare Pautasso, Eduardo Guerra, Hugo Ferreira, Jim Episale, Joe Yoder, and Zishan Rahman for their insights and suggestions during our PLoP Writer's Workshop session.

REFERENCES

Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. 2008. Motivation in Software Engineering: A Systematic Literature Review. 50, 9-10 (2008), 860–878. DOI:http://dx.doi.org/10.1016/j.infsof.2007.09.004

Travis Breaux and Jennifer Moritz. 2021. The 2021 Software Developer Shortage Is Coming. 64, 7 (2021), 39–41. DOI:http://dx.doi.org/10.1145/3440753

Rebecca E. Burnett. 2005. *Technical Communication* (6th ed ed.). Thomson/Wadsworth.

Alistair Cockburn. 2006. *Agile Software Development: The Cooperative Game*. Addison-Wesley.

James O Coplien and Neil B Harrison. 2004. *Organizational Patterns of Agile Software Development*. Prentice Hall. DOI:http://dx.doi.org/0131467409

Edward L. Deci and Richard M. Ryan. 2012. Self-Determination Theory. In *Handbook of Theories of Social Psychology*, Paul A. M. Van Lange, Arie W. Kruglanski, and E. Tory Higgins (Eds.). Vol. 1. SAGE Publications Inc. DOI:http://dx.doi.org/10.4135/9781483346243.n302

Rafael Elizalde and Sussy Bayona. 2018. Interpersonal Relationships, Leadership and Other Soft Skills in Software Development Projects: A Systematic Review. In *Trends and Advances in Information Systems and Technologies*, Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo (Eds.). Vol. 746. Springer International Publishing, 3–15. DOI:http://dx.doi.org/10.1007/978-3-319-77712-2_1

Fabian Fagerholm and Jurgen Münch. 2012. Developer Experience: Concept and Definition. In *2012 International Conference on Software and System Process (ICSSP)*. IEEE, 73–77. DOI:http://dx.doi.org/10.1109/ICSSP.2012.6225984

Brian J. Galli. 2019. Barriers to Effective Communication and Stakeholder Management in Project Environments and How to Overcome These Barriers. 9, 2 (2019), 39–57. DOI:http://dx.doi.org/10.4018/IJAL.2019070103

D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson. 2018. What Happens When Software Developers Are (Un)Happy. 140 (2018), 32–47. DOI:http://dx.doi.org/10/gc8wp7

Aaron Griffith. 2016. Mob Programming for the Introverted. (2016).

Ernest R. Hilgard. 1980. The Trilogy of Mind: Cognition, Affection, and Conation. 16, 2 (1980), 107–117. DOI:http://dx.doi.org/10.1002/1520-6696(198004)16:2<107::AID-JHBS2300160202>3.0.CO;2-Y

International Organization for Standardization. 2018. ISO/IEC 23270:2018(E), Information Technology Programming Languages — C#. (2018).

Riba Maria Kurian and Shinto Thomas. 2023. Importance of Positive Emotions in Software Developers' Performance: A Narrative Review. 24, 6 (2023), 631–645. DOI:http://dx.doi.org/10.1080/1463922X.2022.2134483

Kati Kuusinen. 2016. Are Software Developers Just Users of Development Tools? Assessing Developer Experience of a Graphical User Interface Designer. In *Human-Centered and Error-Resilient Systems Development*, Cristian Bogdan, Jan Gulliksen, Stefan Sauer, Peter Forbrig, Marco Winckler, Chris Johnson, Philippe Palanque, Regina Bernhaupt, and Filip Kis (Eds.). Vol. 9856. Springer International Publishing, 215–233. DOI:http://dx.doi.org/10.1007/978-3-319-44902-9_14

Kati Kuusinen. 2018. Value Creation and Delivery in Agile Software Development: Overcoming Stakeholder Conflicts. In *Global Thoughts, Local Designs*, Torkil Clemmensen, Venkatesh Rajamanickam, Peter Dannenmann, Helen Petrie, and Marco Winckler (Eds.). Springer International Publishing, 123–129. DOI:http://dx.doi.org/10.1007/978-3-319-92081-8_12

Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold P.O.S. Vermeeren, and Joke Kort. 2009. Understanding, Scoping and Defining User Experience: A Survey Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, 719–728. DOI:http://dx.doi.org/10.1145/1518701.1518813

Patrick Lencioni. 2006. *The Five Dysfunctions of a Team*. John Wiley & Sons.

Edwin A. Locke, Karyll N. Shaw, Lise M. Saari, and Gary P. Latham. 1981. Goal Setting and Task Performance: 1969-1980. 90, 1 (1981), 125–152. DOI:http://dx.doi.org/10.1037/0033-2909.90.1.125

Niamh McNamara and Jurek Kirakowski. 2006. Functionality, Usability, and User Experience: Three Areas of Concern. 13, 6 (2006), 26–28. DOI:http://dx.doi.org/10.1145/1167948.1167972

Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå. 2010. A Teamwork Model for Understanding an Agile Team: A Case Study of a Scrum Project. 52, 5 (2010), 480–491. DOI:http://dx.doi.org/10.1016/j.infsof.2009.11.004

Jakob Nielsen. 2012. Usability 101: Introduction to Usability. (2012). https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Don Norman and Jakob Nielsen. 2021. The Definition of User Experience (UX). (2021). https://www.nngroup.com/articles/definition-user-experience/

Daniel Pinho, Ademar Aguiar, and Vasco Amaral. 2023. Patterns for Low-Code Developer Experience. In *Proceedings of the 30th Conference on Pattern Languages of Programs (to be published) (PLoP '23)*. Association for Computing Machinery.

Daniel Pinho, Ademar Aguiar, and Vasco Amaral. 2024. Cognitive Patterns for Developer Experience. In *Proceedings of the 29th European Conference on Pattern Languages of Programs, People, and Practices (EuroPLoP '24)*. Association for Computing Machinery, 1–10. DOI:http://dx.doi.org/10.1145/3698322.3698345

W.G. Poole. 2003. The Softer Side of Custom Software Development: Working with the Other Players. In *Proceedings 16th Conference on Software Engineering Education and Training, 2003. (CSEE&T 2003).* IEEE Comput. Soc, 14–21. DOI:http://dx.doi.org/10.1109/CSEE.2003.1191341

Jeff Sutherland, James O. Coplien, Lachlan Heasman, Mark den Hollander, and Cesário Oliviera Ramos. 2019. *A Scrum Book: The Spirit of the Game*. Pragmatic Bookshelf. http://scrumbook.org

Received May 2024; revised September 2024; accepted February 2025