codurance

Compass

by Codurance

Prepared for:

Compass Test

Detailing:

Codurance

Prepared on:

09/09/2020

The Codurance Compass

Thank you for taking the time to complete the Codurance Compass. Software Craftsmanship is at the heart of Codurance, and these values and the practices that flow from them are how we can continuously deliver great software. Now we have taken these same values and used them to assess the answers you gave and provide some practical advice you can take away and apply in your own organisation.

This report is divided into five categories that we believe capture the essence of an effective software delivery organisation. The categories are summarised on the next page, and then we drill deeper into each one in turn. Hopefully, in the detailed analysis of each category, you will be able to recognise elements of the current situation at your organisation and be able to find some actionable suggestions which you can use to improve.

London / Manchester

Phone: +44 207 490 2967
Email: david.hall@codurance.com

Barcelona

Phone: +34 937 82 28 82
Email: jack.coleman@codurance.com

Your Assessment

Organisational Maturity

Effective software teams consist of aligned stakeholders contributing to the delivery effort from the beginning. They produce well-designed systems that are easy to change and evolve over time. They are focused on the highest value features and can repeatedly deliver effectively across multiple projects.

Continuous Delivery

Good software teams are able to regularly and reliably release valuable product increments. They make use of highly automated processes and feedback loops that allow problems to be found and corrected easily.

People & Culture

Transparency is a prerequisite to good decision making. It builds trust within teams. A culture of learning, along with defined career paths, increases staff retention and helps to drive recruitment.

Effective Team

Strong teams use XP practices to ensure low defect rates and to drive flexibility in changing circumstances. Sustainable pace stops delivery pressure from driving down quality, and leads to better staff retention.

Technical Practices

Teams with a broad range of experiences and backgrounds provide a range of perspectives and increase the likelihood of innovation. Strong, autonomous teams collaborate well - they can cover for each other and are resilient when key members are not available.

Contents

Organisational Maturity

Effective software teams consist of aligned stakeholders contributing to the delivery effort from the beginning. They produce well-designed systems that are easy to change and evolve over time.

They are focused on the highest value features and can repeatedly deliver effectively across multiple projects.

> **The modern business should value useful outcomes over the processes used to get to them and should optimise their delivery capability for customer value above all other considerations.**

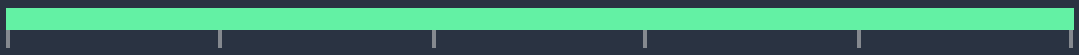
James Birnie
Principal Craftsperson, *Codurance*

Your Responses

DevSecOps

Operations and security stakeholders are actively engaged throughout the development lifecycle.

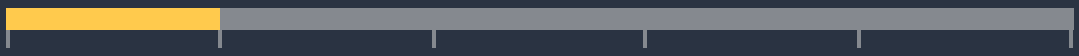
Strongly Agree



Delivering Value

Decision making for IT product and projects is based on what will carry the most value for the business.

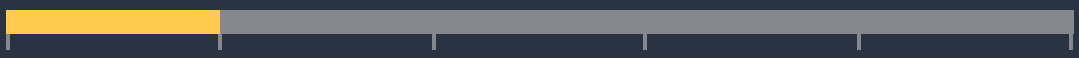
Strongly Disagree



Technical Debt

We measure the accumulation of technical debt, with technical debt prioritised alongside new feature development within the same backlog.

Strongly Disagree



Well Defined Methodology

We have a well-defined methodology for releasing new features to market which involves all areas of the business.

Agree

It is likely that your operations and security teams are involved in your software development life cycle, but they are still treated as external stakeholders by your development teams.

Organisational politics probably has some influence on decision making across the development process. Some decisions are not being made in the best interest of the wider organisation, but rather in the best interest of individuals, departments or teams.

You may find that technical debt levels are continuing to grow and that this is starting to have a direct impact on effective delivery. However, you are unable to bring it under control owing to schedule pressure, the necessary skills or difficulty quantifying the harm it is doing to the organisation to justify the time and effort.

Key requirements from external stakeholders often arrive late, from the development team's perspective, and are sometimes misunderstood. Unfortunately, the only thing that is predictable about development lead times is that they are currently too long.

You are still able to produce good work, but not in a sustainable fashion that keeps all stakeholders happy over the long term. Despite starting promisingly with quick initial progress, defect counts in your products are starting to rise thanks to increasing demand for features and misinterpreted requirements. This is starting to impact on the perception of your products which, while are currently performing well, could be vulnerable to disruption from more agile competitors in the near future.

Try combining people with responsibility for customers, development, security and operations into cross-functional delivery teams. Ensuring that all of the knowledge and skills required are within the team will eliminate delays in acquiring information and significantly reduce the risk of miscommunication as teams build a shared understanding together.

Focus on discovering the value of features and projects as early as possible. Ensure that you have actionable metrics which you can use to measure the value delivered and use that feedback to inform further development. Try forming hypotheses about the potential value of features and use experiments to test them.

Provide training and coaching in practices like continuous re-factoring and simple design to your developers to help slow or eliminate the creation of defects and reckless technical debt. Get used to controlling the scope of products, features and releases to achieve desired time-scales, rather than letting schedule pressure force quality down and technical debt up.

Continuous Delivery

Good software teams are able to regularly and reliably release valuable product increments.

They make use of highly automated processes and feedback loops that allow problems to be found and corrected easily.

> Trust energizes participants. We feel good when things work smoothly. We need to be safe to experiment and make mistakes.
>
> We need testing to bring accountability to our experimentation so that we can be sure we are doing no harm.

Kent Beck
Productive Irritant, *Gusto*

Your Responses

Deployment Cadence

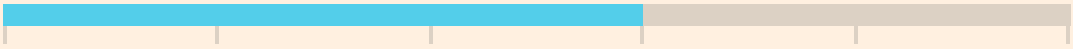
We deploy working code to a production-like environment.

Monthly

Rework

We rarely have to rework deployed code to fix bugs and issues.

Neither Agree Nor Disagree



Automated Pipeline

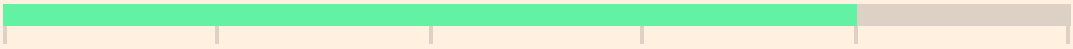
Developers can deploy to a production-like environment using a fully automated pipeline.

Disagree

Confidence to Develop Without Side Effects

We can confidently make changes to our entire codebase, without the risk of unexpected side-effects.

Agree



Your deployment processes are either entirely or mostly manual, and that there tends to be a blurred line of accountability between operations and development teams. Deployments to non-production environments are often automated, but production deployments remain manual.

A large percentage of development time is spent reworking defects that escape into the production environment, pulling developers' attention away from newer, valuable components and features. This leads to risk aversity amongst developers owing to unpredictable side effects.

Operational problems owing to human error are commonplace - important things may get missed or omitted owing to fuzzy accountability between development and operations teams.

Deployment issues and rollbacks typically lead to a lower deployment cadence and slower time to market. When promised features fail top appear to clients, reputational damage follows and clients may look to other vendors to meet their needs.

The combination of defect resolution and slow feature development lead times inevitably increases the cost of getting new features to market owing to fear-based over-engineering and risk-averse behaviour in development teams. This is a typical cause of developer frustration and staff turnover.

Spend some time focusing on deployment and delivery automation, especially for production deployments. You should make sure that all environments replicate your production environment as closely as practical. Choose a continuous delivery strategy that is appropriate to your current situation and implement it in collaboration with your development and operations teams. You should be aiming for the ability to both deploy easily and to recover easily if something goes wrong.

In addition, every time you experience defects, issues or other failures you should be performing an incident retrospective with a focus on root cause analysis. The outcome should result in a fix of the root cause so that it cannot occur again. Make sure that these issues and fixes are logged and visible.

Train and mentor your developers in Extreme Programming practices in order to minimise the risk of defects being introduced in the first place. Demonstrate to them how a good suite of unit, acceptance and smoke tests can give them the confidence they need to make required changes. You should foster and encourage a culture of technical quality, driven by tests.

People & Culture

Transparency is a prerequisite to good decision making. It builds trust within teams.

A culture of learning, along with defined career paths, increases staff retention and helps to drive recruitment.

> Before you are ready to deploy a scientific approach to improving performance, you must first understand and develop your culture.

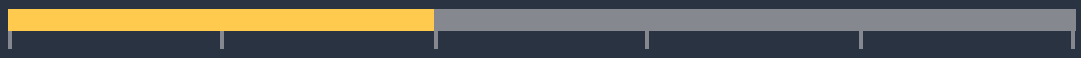
Nicole Forsgren, Jez Humble, and Gene Kim
Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

Your Responses

Transparency

We are an open, transparent organisation which shares all information internally.

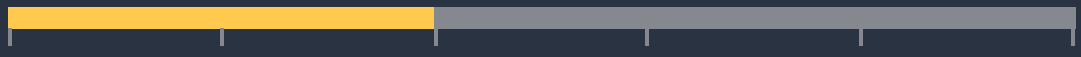
Disagree



Learning

We have a well-established culture of learning and the company invests in educating its employees.

Disagree



Failure Is an Opportunity to Learn

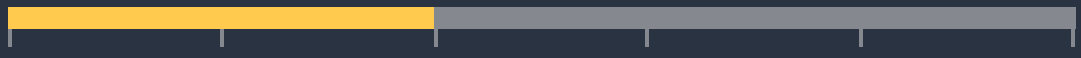
We are open and honest about our failings and treat them as an opportunity to learn, without assigning blame.

Strongly Agree

Career Path

We have a well-defined career path and employees are given the support and guidance they need to grow within the organisation.

Disagree



It seems that there may be a broadly-accepted desire to be transparent in your organisation, but some execution is lacking (perhaps for historical reasons). Typically in this scenario, we find a request-response model of information flow which requires employees to actively seek out information.

Team skill levels may be holding your organisation's progress back and a restrictive training budget may exacerbate this situation. An expectation or reliance on staff members learning in their own time will also hold teams back from achieving to their maximum potential.

There is generally no direct punishment for failure, but neither are failures examined and treated as learning opportunities.

Career paths may be defined and are perhaps highly competitive. Career opportunities may be limited by the need for a position to open up and individual progression may be capped for those who are not interested in moving into management positions.

A sense of frustration over the difficulty of obtaining supposedly open information leads to employees simply not bothering to search for it. This can lead to mistakes owing to misinformed people and non-deliberate information hiding. Information sharing requests may at best be treated with low priority, while at worst they may fall on deaf ears.

A lack of training leads to a lack of specialists and a general sense of lethargy regarding self- and organisational improvement. Highly driven employees, lacking assistance to help them achieve mastery, will disengage and seek out alternative employment at organisations that have a culture of learning.

A culture of hiding or misrepresenting failures (individual and organisational) results in the same mistakes being repeated over and over, when a culture of honest feedback and learning would have corrected the root cause.

One of the greatest dangers of a glass ceiling (especially for those who do not wish to pursue careers in management) is the loss of your most senior people. If it is easier to progress by leaving the organisation then your competitors will gladly hire your best technical staff members.

Find transparency bottlenecks and actively remove them - speak to your employees to understand what is blocking their access to information. Work on moving towards a publish-subscribe model of information dissemination.

Set aside budget for training and optimise and encourage a culture of learning. Consider kicking off initiatives such as lunch-and-learns, open spaces, unconferences and company-supported learning time. Make sure that your training initiatives are directed towards learnings that improve both employee knowledge and capability..

Put an emphasis on incident retrospectives and root-cause analysis. Make sure that you have constructive feedback mechanisms in place that emphasise learning and continuous improvement rather than blame and punishment.

Clarify the career development path for every role in your organisation. Make sure to remove progression bottlenecks and to understand each individual's career goals and aspirations. Ensure that your progression plans allow technical people to progress as far as non-technical people.

Effective Team

Teams with a broad range of experiences and backgrounds provide a range of perspectives and increase the likelihood of innovation.

Strong, autonomous teams collaborate well - they can cover for each other and are resilient when key members are not available.

> Our research shows that three things are highly correlated with software delivery performance and contribute to a strong team culture: cross-functional collaboration, a climate for learning, and tools.

Nicole Forsgren, Jez Humble, and Gene Kim
Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

Your Responses

Diversity

We are a diverse team with people from a wide range of backgrounds and experiences.

Disagree

Autonomy

Teams have autonomy to decide on the tools, technologies, practices and methodologies they use to complete their tasks.

Strongly Disagree

Whole Team

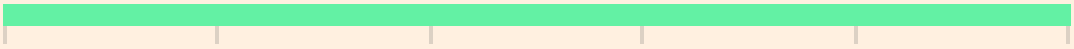
The team have immediate access to all the people and knowledge they require to complete their work.

Neither Agree Nor Disagree

Knowledge Sharing

Knowledge is shared at all levels throughout the team and each member can perform any required task.

Strongly Agree



Your organisation is trying to work in a more collaborative fashion. Where functional teams exist they work together in a spirit of cooperation towards shared goals and silos are rarely a problem. You may also have started to build cross-functional teams for specific products, but they still might be struggling to shake the functional organisation mindset or may still be dependant on a small number of specialists in other teams.

Your teams have some flexibility in choosing ways of working, tools and technologies but are probably still tightly constrained in at least one of these dimensions. They also have access to the information they need to do their work, but often this access is not immediate. Technical knowledge is openly shared within your teams but domain knowledge is not as widely shared.

You probably have some constraints in place that are preventing teams from using the best tools for the job or are compelled to continue to use outdated technologies in some circumstances. This is probably impacting both their productivity and their ability to innovate. It could well also be having an impact on morale as individuals lack autonomy in their working environment and possibly worry about falling behind industry trends in technology and tooling. Unfortunately this situation is probably having a visible impact on staff retention and makes recruiting new team members more difficult.

While your teams do, eventually, have access to all the information they need to do their work they may find themselves delayed by having to go through gatekeepers that slow this access down or introduce noise into the signal that the team is after. This can cause significant delays if it is not possible for teams to hold direct conversations with knowledgeable stakeholders and instead have to work through a gatekeeper. Your teams are probably also struggling with other constraints on access to important stakeholders, such as their overall availability or their location. This results in domain experts becoming a constraint on value delivery, and the loss of these experts is a significant organisational risk.

We recommend delegating full authority on tooling, process and technology to the people who are best qualified to make those decisions. Provide budgetary constraints if absolutely necessary, but trust your team members to make the right decisions.

Start forming cross-functional teams (if you have not already) and ensure that they are truly cross-functional by including all the skills necessary for them to deliver their product. This will probably involve adding competencies to some teams that are not normally considered part of a software development team, such as marketing, customer support or graphic design as well as more standard roles like developer, QA and system operations.

Facilitate rapid access to stakeholders for teams. Team members should be able to speak directly with whoever they need to, from customer to CEO, without going through a gatekeeper or mediator. The goal is to reduce delays and rework by ensuring teams have access to people and information on demand.

Facilitate knowledge sharing sessions, run by domain experts, for your teams. These can include workshops, subject matter training, shadowing and introductions to techniques like Domain Driven Design. Work to turn development teams into domain experts in order to reduce their dependence on external knowledge sources, and to increase their capability to act autonomously.

Technical Practices

Strong teams use good technical practices to ensure low defect rates and to drive flexibility in changing circumstances.

Modular, testable architecture combined with clean code helps so simplify the effort and risk associated with change.

> A programming language is a medium for communicating methods, not just a means for getting a computer to perform operations - programs are written for people to read as much as they are written for machines to execute.

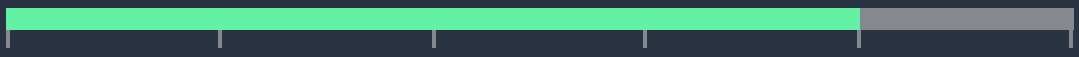
Harold Albelson and Gerald Jay Sussman
Lisp: A Language for Stratified Design

Your Responses

Test-Driven Development

We value Test-driven Development (TDD) and never write production code without first writing a test.

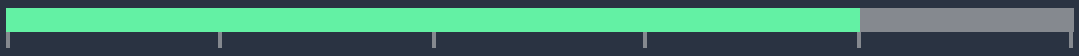
Agree



Modular Architecture

Change is localised and tested with minimal impact on other parts of the application.

Agree



Clean Code

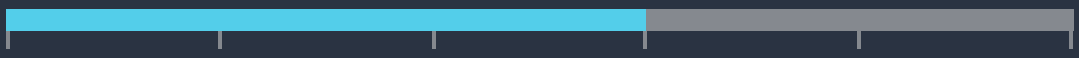
Our code is easy to understand and maintain.

Agree

Peer Review

All production code is peer-reviewed comprehensively before it is considered “done”.

Neither Agree Nor Disagree



It appears that your organisation has made an effort to adopt good technical practices but that currently adoption is uneven and inconsistent and/or you a struggling to kick on to the next level in order to start seeing the big payoffs. While test-driven development is used sporadically, it's probably more common for teams to write unit tests after writing production code and most legacy code is probably still dependent on manual acceptance testing. Your teams are refactoring, but rather that refactoring as they go it's likely that tend to allow technical debt to accumulate for a while before tackling it in a large batch.

Your team may understand modular architect and clean code at a high-level but often unable to use that to guide them in refactoring and developing new features. There is likely a peer review process in place, but peer reviews are sporadic and/or inconsistent across the organisation. Generally speaking, your developers do not pair program unless the task at hand is especially complex or time critical.

Things are going relatively well, but you are coming under increasing pressure to reduce the amount of time it takes to bring features to market as opportunities are being missed and its taking too long to get feedback from customers.

While product quality is generally pretty good, you have a nagging feeling that defects are making it to production more frequently than they should. When this happens, the time to fix can be long which damages the customer's perception of your products and your organisation. It's possible that, in an attempt to fix these issues, there is a large QA effort prior to release to catch defects. This QA effort is expensive due to the number of people involved and has the unfortunate side effect of increasing time to market due to the long cycle time of the QA process.

We recommend defining a test automation strategy, the goal of which should be to replace the vast majority of any existing manual testing. This should be achieved with appropriately scoped automated tests and a re-focus of manual testing on inventive, creative tasks like test design and exploratory testing.

Change the emphasis from training developers on the practices of test-driven development, modular architecture, clean code, refactoring, simple design and pair programming to coaching your developers to effectively employ these practices in their day to day work in their current projects. Build a shared understanding of what "good" looks like in your organisation, so that the same standards are applied across your organisation when pair programming or other forms of peer review.

If your organisation has a large legacy code base it would be wise to develop some architectural principles and possibly a redesigned architecture to give your development teams a target to aim for when they are refactoring.

Continue to strive for continuous improvement. When defects are discovered and fixed in production be sure to conduct a retrospective and use root cause analysis to try to uncover the true cause of the defect being shipped. You should then take actions to improve process and practices to prevent a re-occurrence.

Summary

It is important to note that these results are high-level views.

Our recommendation for your next steps would be to circulate and discuss your results internally to reach a collected consensus.

Speaking to one of our advisors, we can then generate a strategy in improving your business and it's evolutionary potential moving forward.

London / Manchester

Sales Contact: **David Hall**

+44 744 70 62 036
david.hall@codurance.com
codurance.com

3 Sutton Lane, 3rd floor
London, EC1M 5PU

2 Mount Street
Manchester, M2 5WQ

• • •

Barcelona

Sales Contact: **Jack Colemanzo**

+34 937 82 28 82
jack.coleman@codurance.com
codurance.es

Carrer de Pallars 99,
4th floor, room 41
Barcelona, 08018

Key Takeaways