Architecting a Globally Distributed Software Organization for Continual Development

Anastas Stoyanovsky
(Watson Information Retrieval on Elasticsearch)
(Watson Information Retrieval Engine)
Watson Discovery Service
Red Hat OpenShift + IBM Cloud Pak for Data
#wirehose
REMOTE
Someone else’s repository.

Fork!

REMOTE
Your fork of the repository.

cool_repo

cool_repo_fork

Clone to your computer from GitHub.

REMOTE
Push and Pull to your fork ‘origin’.

PULL from ‘upstream’ changes to original.

LOCAL
Use your computer’s terminal to talk to two repositories via two remotes to the GitHub servers.
SOME TIME LATER...
= IBM Data & AI
Red Hat OpenShift + IBM Cloud Pak for Data
Literature Review:
Global Development

- Face to face initial meeting
- Daily handoffs
- Shared infrastructure
  - Source repositories
  - CI/CD infrastructure
  - Artifact management
  - &c.
- Shared expectations for development practices
Literature Review: Global Development

- **Face to face initial meeting**
- Daily handoffs
- Shared infrastructure
  - Source repositories
  - CI/CD infrastructure
  - Artifact management
  - &c.
- Shared expectations for development practices
SOME TIME LATER...
SOME TIME LATER...
SOME TIME LATER...
3h

(all images to scale)
ADR N: Brief Decision Title

Context goes here.

Describe the forces at play, including technological, political, social, and project local. These forces are likely in tension, and should be called out as such. The language in this section is value-neutral. It is simply describing facts. The context provides background context so we'll understand the rationale for the decision.

Decision

This section describes our response to these forces. It is stated in full sentences, with active voice. "We will ..."

Status

choose one: [Proposed | Accepted | Deprecated | Superseded]

if deprecated, include a rationale.

If superseded, include a link to the new ADR

Consequences

Describe the resulting context, after applying the decision. All consequences should be listed here, not just the "positive" ones. A particular decision may have positive, negative, and neutral consequences, but all of them affect the team and project in the future. There are always consequences to a decision. Shoot for at least 2-3 as a starting point.
Literature Review: Global Development

- Face to face initial meeting
- Daily handoffs
- Shared infrastructure
  - Source repositories
  - CI/CD infrastructure
  - Artifact management
  - &c.
- Shared expectations for development practices
Literature Review: Global Development

- Face to face initial meeting
- Daily handoffs
- Shared infrastructure
  - Source repositories
  - CI/CD infrastructure
  - Artifact management
  - &c.
- Shared expectations for development practices
Shared Expectations

Forming

Norming

Performing

Team Effectiveness

Team Performance
synthesizing
Synthesized Way of Working

(a starting point)

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- "Atomic" issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Effective tactic #1: A minimal set of architecture diagram types

Component/Connector Diagrams

Sequence Diagrams
Effective tactic #1: A minimal set of architecture diagram types

Component/Connector Diagrams

Sequence Diagrams

Goals:

1. Maintain shared understanding
2. Minimize miscommunication
3. Minimal time and effort burden
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Effective tactic #2: Distributing design authority using ADRs

Benefits:

1. Concisely record research and/or discussion
2. Asynchronous communication
3. Evidence-based disagreement (“context”, “consequences” sections)
4. Encourages and habitualizes critical thinking
5. Shared understanding - minimize “surprise factor”
6. Accountability to stick to decisions
7. Normalizes autonomy
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
  ✓
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Synthesized Way of Working

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Effective Tactics: Recap

1. Minimal but sufficient architecture diagramming
2. Distribute design authority through ADRs
Development practices
Communication and Collaboration
WATERFALL vs. AGILE

Waterfall Process:
1. Discovery
2. Planning
3. Design
4. Development
5. Testing
6. Deployment
7. Maintenance

Agile Process:
1. Planning
2. Design
3. Development
4. Testing
5. Deployment
6. Maintenance

The diagram illustrates the differences between the Waterfall and Agile methodologies in project management.
Development Practices

- "Atomic" issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Development Practices

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming ✔
- Distributing design authority via ADRs ✔
- Functional testing???
Effective tactic #3: Adoption through demonstration of value
Effective tactic #3: Adoption through demonstration of value
Development Practices

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming ✅
- Distributing design authority via ADRs ✅
- Functional testing???
Development Practices

- “Atomic” issues
- Daily handoffs
- Minimal but sufficient diagramming
- Distributing design authority via ADRs
- Functional testing
Development practices
Communication and Collaboration
Development practices
Communication and Collaboration
“20x5” Development!

🌶🌶🌶

Continual, global development
“20x5” 24x5 Development! 🌶🌶🌶
“20x5” 24x5 Development!
What has been done

- Implemented widget A
- Made widget A work with widget B

What needs to be done

- Update widget C interface to accept an instance of widget A to do task X
- Make task X not always preceded task Y
What has been done

- Implemented widget A
- Made widget A work with widget B

What needs to be done

- Update widget C interface to accept an instance of widget A to do task X
- Make task X not always preceded by task Y
Congratulations!

This is the first issue in your demo project. This issue represents the effort that went into launching the app you're working with right now.

You may have already noticed that this issue is assigned the Fixed state. As the application is already up and running, there's nothing more for you to do. However, there are lots of other issues in this project that are assigned the Open state and are waiting for your input.

To continue along your journey, click through to Create a demo project.

+ Link issue

Write a comment, @mention people

Lena Pishkova about 1 month ago
We've added a few work items to this issue that represent what steps were taken to resolve this issue. It took a little more time than expected...
Effective tactic #4: TCES format

- Today: implemented widget A and got it working with widget B
- Current problems: task X can't use widget A yet, and tasks X and Y are coupled
- External concerns: waiting on latest update of dependency W
- Stakeholders: @implementationPartnerInTheOtherCountry @personWhoImplementedTasksXandY

- personWhoImplementedTasksXandY 7 minutes ago
  they have to be coupled that way because of P, but if it needs to use A then you could probably do Q

- implementationPartnerInTheOtherCountry 2 minutes ago
  makes sense, I'll try to go after that today
Effective tactic #4: TCES format

- **Today:** implemented widget A and got it working with widget B
- **Current problems:** task X can't use widget A yet, and tasks X and Y are coupled
- **External concerns:** waiting on latest update of dependency W
- **Stakeholders:** @implementationPartnerInTheOtherCountry @personWhoImplementedTasksXandY

---

- **personWhoImplementedTasksXandY** 7 minutes ago
  - they have to be coupled that way because of P, but if it needs to use A then you could probably do Q

- **implementationPartnerInTheOtherCountry** 2 minutes ago
  - makes sense, I'll try to go after that today
Effective tactic #4: TCES format

- **Today:** implemented widget A and got it working with widget B
- **Current problems:** task X can't use widget A yet, and tasks X and Y are coupled
- **External concerns:** waiting on latest update of dependency W
- **Stakeholders:** @implementationPartnerInTheOtherCountry @personWhoImplementedTasksXandY
Effective tactic #4: TCES format

- **Today**: implemented widget A and got it working with widget B
- **Current problems**: task X can't use widget A yet, and tasks X and Y are coupled
- **External concerns**: waiting on latest update of dependency W
- **Stakeholders**: @implementationPartnerInTheOtherCountry @personWhoImplementedTasksXandY

2 replies

- **personWhoImplementedTasksXandY** 7 minutes ago
  - they have to be coupled that way because of P, but if it needs to use A then you could probably do Q

- **implementationPartnerInTheOtherCountry** 2 minutes ago
  - makes sense, I'll try to go after that today
Effective tactic #4: TCES format

- **Today**: implemented widget A and got it working with widget B
- **Current problems**: task X can't use widget A yet, and tasks X and Y are coupled
- **External concerns**: waiting on latest update of dependency W
- **Stakeholders**: @implementationPartnerInTheOtherCountry @personWhoImplementedTasksXandY

- **personWhoImplementedTasksXandY** 7 minutes ago
  - they have to be coupled that way because of P, but if it needs to use A then you could probably do Q

- **implementationPartnerInTheOtherCountry** 2 minutes ago
  - makes sense, I'll try to go after that today
TCES Format

Benefits

- Concise holistic daily updates *(Today)*
- Push-based information flow instead of pull-based *(Stakeholders)*
- Descriptive over prescriptive *(Current problems)*
- Normalizes cross-team collaboration over siloing / escalation *(External concerns)*
- Replaced 30+ min daily morning status/planning meeting
Effective tactic #4: Asynchronous TCES handoff mechanism
Development practices
Communication and Collaboration
Development practices
Communication and Collaboration

Team Effectiveness

Forming
Storming
Norming
Performing

Team Performance
1. Minimal but sufficient architecture diagramming
2. Distribute design authority through ADRs
3. Adoption through demonstration of value
4. TCES handoff mechanism
Emergent practices
Functional (Component Level)
Unit
Integration
E2E
Unit

Functional (Component Level)

Unit
Effective tactic #5: Inverting the testing pyramid
Why?

- Confidence in comprehensive test suite sufficed?
- Differences in unit test organization increased barrier to entry?
- Observed that refactoring became easier for everything but the deep internals?
Inverting the testing pyramid: Benefits

- Rapid large-scale refactoring
- Rapid iteration
- Lower cost to false starts / design changes
- Confidence in input/output remains
Effective Tactics: Recap

1. Minimal but sufficient architecture diagramming
2. Distribute design authority through ADRs
3. Adoption through demonstration of value
4. TCES handoff mechanism
5. Inverting the testing pyramid
We made it!

Team Effectiveness vs. Team Performance diagram showing the stages of team development: Forming, Storming, and Norming, leading to the Performing stage.
(executable specifications)
Functional (Component Level)
Unit
Integration
E2E
AHEAD OF SCHEDULE,
UNDER BUDGET
Receive Query → Generate Initial Results → Model Inference → Reorder results

Feature Generation
Receive Query

Generate Initial Results

Feature Generation

Model Inference

Reorder results

≈300ms
Tech Debt Portfolio as Company Grows

- Decision debt
- Technical Product debt
- Developer efficiency debt
- Security debt
- Stability debt
- Maintenance debt

Company Growth Stage:
- Traction
- Inflection
- Scale
- Expansion

Tech Debt
**Paraphrased Conway's Law**

The structure of software will mirror the structure of the organisation that built it for example.

- **Organisation**
  - Small distributed teams
  - Large colocated teams

- **Software**
  - Modular, service architecture
  - Monolithic architecture
Conway's Law

The structure of software will mirror the structure of the organisation that built it for example.

**Organisation**
- Small distributed teams → modular, service architecture
- Large colocated teams → monolithic architecture

**Software**
Effective tactic #6: Fast-forwarding Conway’s law
Effective tactic #6: Fast-forwarding Conway’s law
Effective tactic #6: Fast-forwarding Conway’s law
Effective tactic #6: Fast-forwarding Conway’s law
CSPU: ConcUrent Staged Pipelines

CUSP is a framework for constructing and executing pipelines. It represents a pipeline as a directed graph with a single source and sink, constructed using JGraphT, executed using ParSeq, and visualized using tools from both of those projects.

[https://github.com/IBM/cusp](https://github.com/IBM/cusp)

**Effective tactic #6:** Fast-forwarding Conway’s law
Effective Tactics: Recap

1. Minimal but sufficient architecture diagramming
2. Distribute design authority through ADRs
3. Adoption through demonstration of value
4. TCES handoff mechanism
5. Inverting the testing pyramid
6. Fast-forwarding Conway's law
Effective Tactics: Recap

1. Minimal but sufficient architecture diagramming
2. Distribute design authority through ADRs
3. Adoption through demonstration of value
4. TCES handoff mechanism
5. Inverting the testing pyramid
6. Fast-forwarding Conway's law
EPILOGUE
Thank you!
Acknowledgements

**US team**
- Charles “Chuck” Gala
- Chris Nolan
- Jennifer Manning
- Will Chaparro (*dev manager*)
- Yi-Shiuan Tung

**Tokyo team**
- Daiki Tsuzuku
- Hiroaki Kikuchi (*lead*)
- He Yanxia
- Takahito Tashiro
- Takashi Fukuda
- Tominaga Yasuyuki

**Shepherd**
- Yulia Pieskova
Thank you!