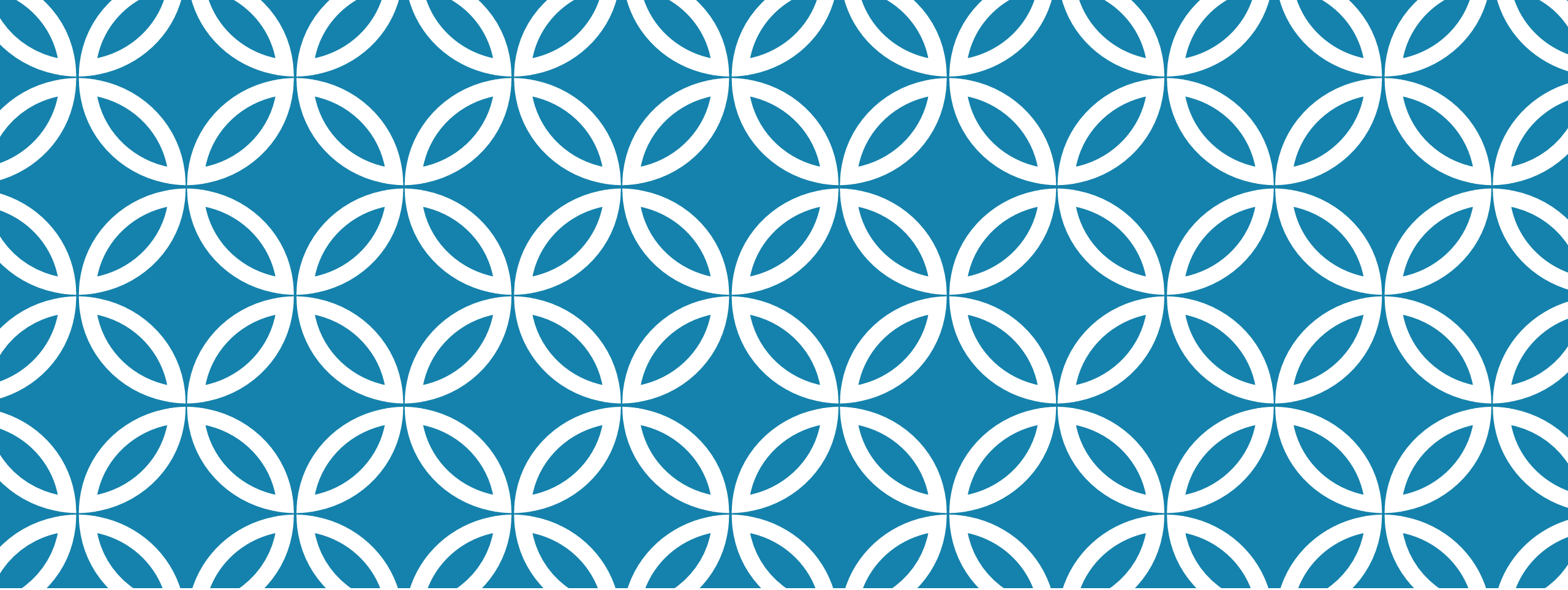


ITMO 430

Week 02 – The 3 Ways
Chapter 1-4

OBJECTIVES

Understand the 3 ways of DevOps

Understand how to implement the 3 Ways in development, operations, and business practices

Discover and learn about tools used in the process of implementing the 3 ways

OUTCOMES

At the conclusion of this week's lab and lecture you have an understanding of the 3 Ways and how they can be applied in development, operations, and business objectives. We will also take a survey of tools that will help us implement these practices.

REVIEW

Flow

- Make work flow by making it visible

Fast and constant Feedback

- Automated processes
- Testing and frequent deployment

Continual learning and experimentation

- High trust
- Anti-fragile
- Supports learning and experimentation
- Amplify Feedback loops

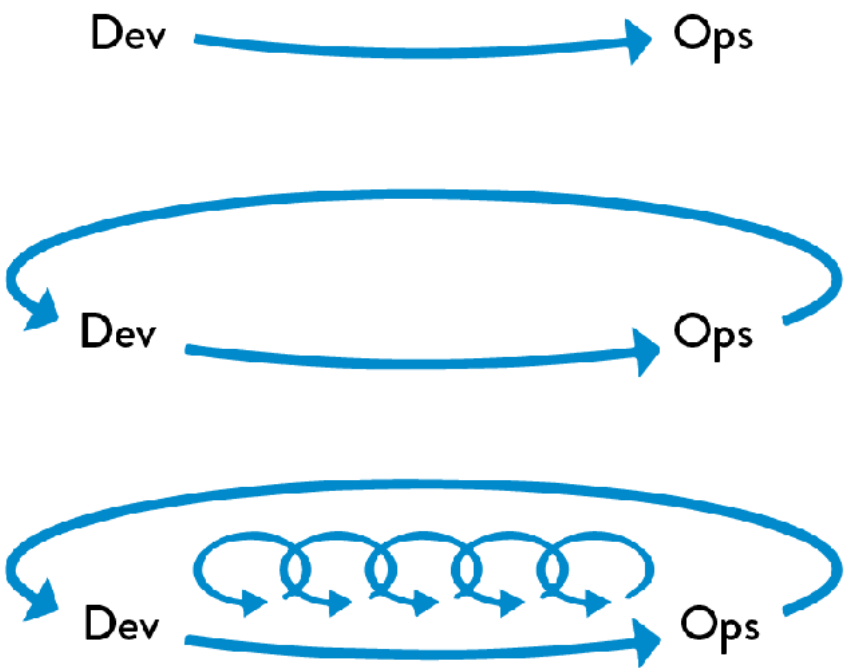
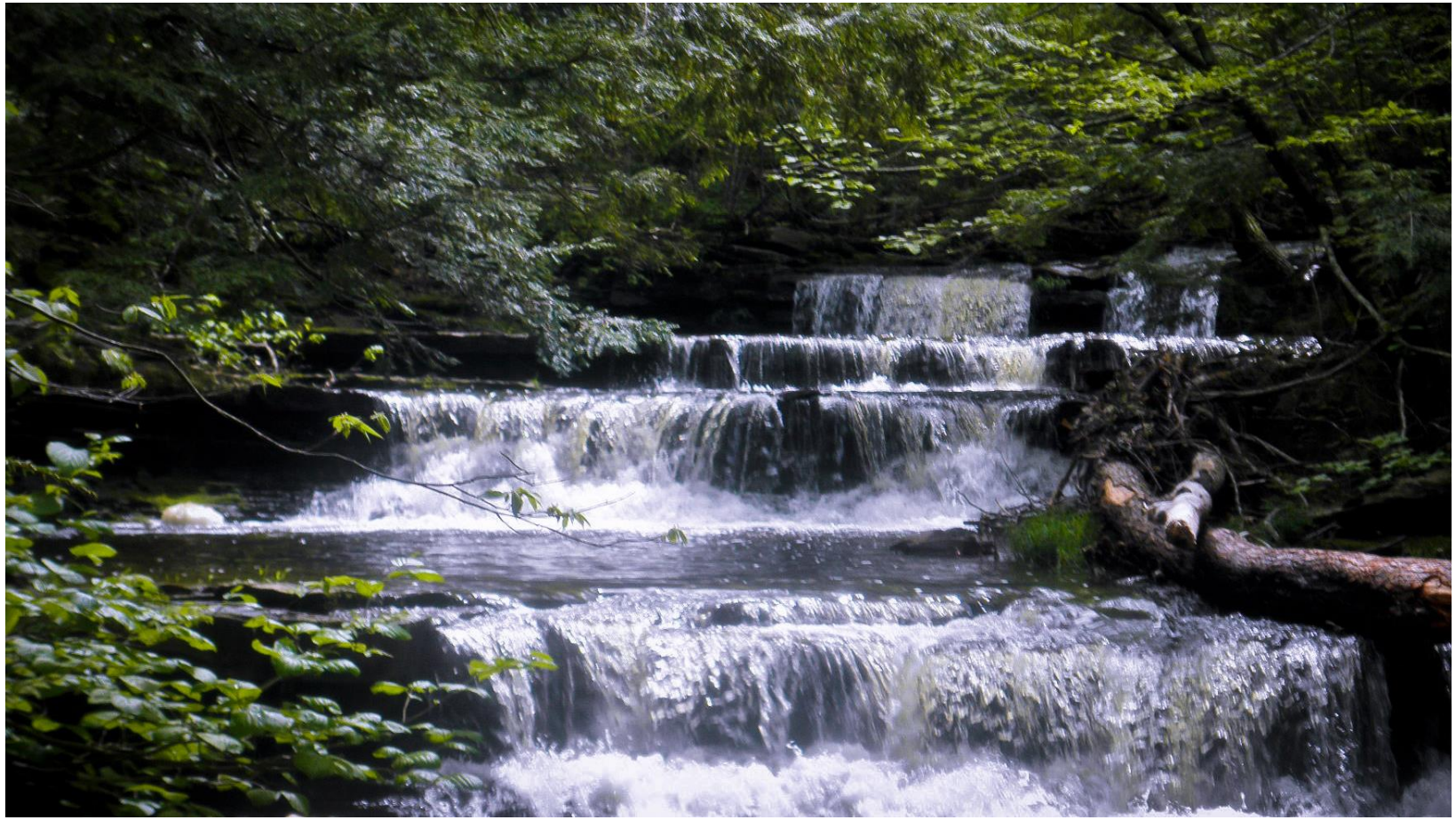


Figure 5: *The Three Ways (Source: Gene Kim, "The Three Ways: The Principles Underpinning DevOps," IT Revolution Press blog, accessed August 9, 2016, http://itrevolution.com/the-three-ways-principles-underpinning-devops/.)*

PRINCIPLES OF FLOW – THE 1ST WAY

Value Stream

- From “Aha!” to “$$$$”
- Streams only flow one way



PRINCIPLES OF FLOW – THE 1ST WAY

Work flows from developers to operations

Because of this we need to optimize for global goals not local

- "Hey my part was finished…your part is not my problem!"
- Make work visible (use some agreed upon product)
  - No emailing spreadsheets! Ever!
  - Reduce lead times (from raw material to finished goods in the hand of the customer)
  - Decrease the amount of time needed to deploy changes

PRINCIPLES OF FLOW – THE 1ST WAY

Visibility

- Since our work is digital–it is invisible
- It is easy to switch to another task but is also a danger
- Loose productivity in the course of task switching
  - You appear productive but in the end you are just busy and not really productive
- Making work public and the flow of work public helps reveal these issues
  - May also relieve someone of a large burden or undue stress
- The Japanese have a term, “Kanban” 看板
  - Basically means, “billboard or sign-card”
  - Essentially a digital version of a white board with “sticky notes on it”

KANBAN


To Do
In Progress
Done
USE KANBAN
LEARN ABOUT KANBAN
GET SOME STICKY NOTES

KANBAN

Trello

- Founded by Joel Spolsky (founder of StackOverflow) and Michael Pryor
- Started in 2011
- Recently sold the company for $425 million dollars
- Different from Gantt charts and Project Management
  - Better able to reflect changes in work and flow
  - Better tool
  - Miguel De Icaza uses it (celebrity endorsement – so it must be good!)
  - Let's check it out – trello.com

KANBAN ADVANTAGES

Now that flow is visible you can see where things are bottle necked

You can see who might be overloaded

You can see available tasks if you happen to be free

Tasks move from left to right

- To Do, Doing, Done
- Done means in the hands of customers
- Objects (cards) can be moved back for adjustments

WIP

Work In Progress

- This is what kills most projects
- The little things that pop up, “putting out fires”
- Prevents you from focusing on main goal at hand
- Or additional feature requests that are added onto the project
- Good teams limit WIP
  - For instance only allow three WIP tasks to be added
  - No more can be added until those three are finished

REDUCE BATCH SIZE

Kanban helps you reduce your batch size

- You can break tasks down into atomic steps that have a clear state of done or not done
- “Create the website” is not a clear task, when is that done?
- Install Apache2 webserver is a clear task with a done/not-done state
- Look in your text about the envelope stuffing example.

REDUCE HANDOFFS

Enable the correct person with the correct privileges and rights to do their job

If people have to wait for approval

- Eventually they will find ways (usually insecure) to go around that “security” barrier.
- Handoffs reduce first-hand knowledge.
- Every additional team that touches the code in question has to re-learn its purpose.
- Like the Greek General Xenophon and the ancient city of Babylon
  - Once a great city, but by the 3rd century BC – no one knew why it was abandoned or what it even was

DEPLOY TIME

In our DevOps transformation we will:

- Measure our deploy time in minutes, to do this we need:
  - Production Environment – need to work with your group to architect and deploy an environment
  - Code Deployment – Need a seamless automated code deployment (no manual parts)
  - Test Environment – All member need to be able to deploy a test environment locally on their own systems
  - Turtles all the way down

P.24 ELIMINATE WASTE

Partially done work

Extra processes

Extra features

Task switching

Waiting

Motion (effort needed to do something)

Heroics (staying 48 hours in a row by yourself to solve an issue)

Defects (Pokeyoke)

CHAPTER 3 THE 2ND WAY: PRINCIPLES OF FEEDBACK

Complex work is managed so that problems in design and operations are revealed

Problems are swarmed and solved

- Why? Prevent accrual of technical debt

New local knowledge is gained and exploited globally

Leaders then create other leaders to reinforce this pattern

Pushes decisions as close to the work as possible

Software examples

- Git, Github, and Github Issues – see video

CHAPTER 4 THE THIRD WAY PRINCIPLES OF CONTINUAL LEARNING & EXPERIMENTATION

Avoid blame, instead swarm the problem

- When we avoid fixing issues (out of fear or apathy) things don't stay the same they get worse (entropy)
- Reserve development time to pay down technical debt
  - Remember the Linkedin example from the Gene Kim Video
  - Local discoveries become global knowledge
    - Share it
    - Publish it
    - Start a podcast/blog
  - Inject resilient patterns and tools into work
- Have leaders reinforce this learning and experimentation
- Can often be done in something as simple as having team lunch together

CONCLUSION

Time to do great things

- Focus on Flow:
  - Left to right
- Continuous Feedback
  - Small batches
  - Reduce WIP
- Continual Learning and Experiementation
  - Local knowledge -> Global
  - Share
  - Reinforce this
  - Pay down technical debt