

Attacking your queues

This publication forms part of Value, Flow, Quality® Education. Details of this and other Emergn courses can be obtained from Emergn, 20 Harcourt Street, Dublin, D02 H364, Ireland or Emergn, 190 High St, Floor 4, Boston, MA 02110, USA.

Alternatively, you may visit the Emergn website at <http://www.emergn.com/> education where you can learn more about the range of courses on offer.

To purchase Emergn's Value, Flow, Quality® courseware visit <http://www.valueflowquality.com>, or contact us for a brochure - tel. +44 (0)808 189 2043; email valueflowquality@emergn.com

Emergn Ltd.
20 Harcourt Street
Dublin, D02 H364
Ireland

Emergn Inc.
190 High St, Floor 4
Boston, MA 02110
USA

First published 2012, revised 2021 - printed 16 July 2021 (version 2.0)

Copyright © 2012 - 2021

Emergn All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, transmitted or utilised in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without written permission from the publisher.

Emergn course materials may also be made available in electronic formats for use by students of Emergn and its partners. All rights, including copyright and related rights and database rights, in electronic course materials and their contents are owned by or licensed to Emergn, or otherwise used by Emergn as permitted by applicable law. In using electronic course materials and their contents you agree that your use will be solely for the purposes of following an Emergn course of study or otherwise as licensed by Emergn or its assigns. Except as permitted above you undertake not to copy, store in any medium (including electronic storage or use in a website), distribute, transmit or retransmit, broadcast, modify or show in public such electronic materials in whole or in part without the prior written consent of Emergn or in accordance with the Copyright and Related Rights Act 2000 and European Communities (Copyright and Related Rights) Regulations 2004. Edited and designed by Emergn.

Printed and bound in the United Kingdom by Apple Capital Print.

CONTENTS

Introduction 1

1 What is a queue? 2

- 1.1. The effect of queues on cycle time 4
- 1.2. Zero queues 8

2 Why don't we see queues? 10

3 What causes queues? 13

- 3.1. Work 13
- 3.2. Long queues 18

4 Where do we find queues in IT? 20

- 4.1. The fuzzy front-end 20
- 4.2. Specialisms 23
- 4.3. Environments 27

5 Controlling queues 28

- 5.1. Making the queue visible 29
- 5.2. Estimating queues 29
- 5.3. Sequencing 32
- 5.4. Identifying the problem area: bottlenecks 34

6 How do we quantify a queue? 36

- 6.1. When do we do this? 38

7 Conclusion 40

Bibliography 42

INTRODUCTION

'The enemy of flow is the invisible, and unmeasured queues that undermine all aspects of development performance. Stagnant piles of idle work lengthen cycle time. At the same time, they delay vital feedback and destroy process efficiency. Yet today, these queues remain unmanaged. 98% of development teams today neither measure nor control their queues.'

Don Reinertsen, The Principles of Product Development Flow

In this session we will examine the most important and likely cause of delay – not only in software development in general – but in your own work, whether you are looking at your emails or launching a major new product for the company. Queues are found everywhere in our lives and they are the key source of delay. Despite this, they are rarely measured, and therefore also rarely managed.

We will consider why this is; the properties of queues and the damage they cause. We will also consider why queues exist – there are extremely good reasons as to why organisations might want to have a relatively high level of queues. The danger is that they are unaware of the real cost of queues and thus fail to make an informed decision. Following this session, you should be better able to assess what level of queues exist within your organisation and decide whether you should be taking steps to reduce them.

At the end of this session, the student will have an appreciation of:

1. What a queue is.
2. Why queues are invisible.
3. What causes queues.
4. Where queues are found in IT and software development.
5. Methods of controlling queues.
6. Quantifying a queue.
7. When action on queues is necessary.

1

WHAT IS A QUEUE?

If you've ever decided to go to India, whether for business or on holiday, you'll have experienced the fun of getting a visa. Despite the fact the Indian government has significantly improved the process in recent years, it still takes a fair amount of time, from the moment you begin filling in the online form to collecting your passport with the visa pasted inside. Granting the visa (checking the form, printing the visa, attaching it to the passport) probably takes only a couple of minutes. Even filling in the form and gathering your documents together probably takes you only 20 minutes. Most of your time is spent travelling to and from the visa centre, waiting in line to hand over the documents, returning 3 days later to collect everything (having waited in line once more). The point is that the 'queue' is not only the time you spend physically waiting in a queue, but all of the inactive, waiting time. This is true even though much of the inactive time feels busy – cursing the website as your session drops out, travelling, taking a ticket number, finding the correct desk...

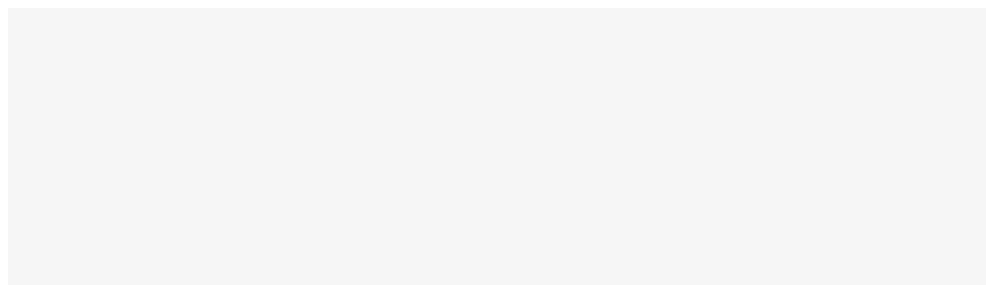
Activity 1: Where is it stuck?

The screenshot shows the 'INDIA VISA' website with the Indian flag and the text 'APPLY FOR INDIA VISA IN UK'. A navigation bar includes 'VISA INFORMATION' and 'ALL ABOUT YOUR VISAS'. Under 'ALL ABOUT YOUR VISAS', there's a list of visa types: Tourist Visa, Business Visa, Urgent Business Visa, Employment Visa, Student Visa, Entry Visa, Long Term Entry Visa (5 Years), Conference Visa, Journalist Visa, Research Visa, Transit Visa, Medical Visa, Medical Escort Visa, and Transfer of Visa. The main content area is titled 'PROCESSING TIME' with a breadcrumb 'HOME > ALL ABOUT YOUR VISAS > PROCESSING TIME'. It features three tabs: 'Know Your Visa', 'Applications by Post - Checklist', and 'Processing Time' (which is highlighted). Below the tabs, text states: 'Processing times vary from application to application at the sole discretion of the High Commission of India and its Consulates in the UK, and some applications may take longer than others to get processed. Hence, confirm your travel plans only after obtaining the visa. Here are some estimates of processing times under normal circumstances:'. A bullet point below explains that applications in London, Birmingham, and Edinburgh take 2-3 working days, while those in Manchester, Cardiff, and Glasgow take 5-7 working days.

This activity will take 5 minutes.

Look at the picture above. You will see that the times for processing Indian Visa applications differ.

Why do people from Manchester, Glasgow and Cardiff have to wait twice as long as people in the other three locations?



Commentary:

It is common to organise operations with several front offices, which are served by a single back office. In this case, the evidence suggests that applications are taken from the three cities either to London, or a location near London. This means time spent waiting in an internal transfer queue. Although the activity time is the same, the process will be much slower for the three Consulates further from the back office where the applications are being dealt with.

1.1. The effect of queues on cycle time

All of our working life is filled with queues. Because we are busy, it can feel quite odd when we realise that of the total length of time a project takes, very little is actually work. A project spends most of its time in a series of queues. The queues range from really big and obvious delays, (waiting to have a team assigned to the project); to minor and almost untraceable, (a request for information in an email sitting unanswered in a colleague's inbox).

Activity 2: Waiting emails?

How often do you use email? Most of us now carry smartphones around that mean we can check email even when on the move. It has become one of the main ways we communicate. It is also a good example of a queue that we deal with on a daily basis.

This activity will take 15 minutes.

Open your inbox and look at the list of emails. Select a few outstanding requests or tasks from the email list at random. They don't have to be new or unread, simply not yet actioned. As long as they are in your inbox they are still waiting to be dealt with. Most of our emails are waiting, either because we've decided not to respond to them immediately, or more commonly, because they actually require us to do some work, and of course that work often depends on somebody else.

How long, on average, are your emails waiting since their arrival? (e.g. It has been in the inbox for 3 days).

How long would it take to actually deal with these requests? (e.g. It will take me 5 minutes to carry out the action in the mail).

Try to answer these questions by filling in the following table:

Email subject	Days in inbox	Estimated action time

Commentary:

In our experience, the time needed to do something about emails that are waiting for action is much shorter than the time that they sit waiting.

Our emails often build up not because they take a long time to respond to – once you have the right information, writing up the few sentences is quick – but because you are waiting for that information. It is the waiting that creates the queue and slows the process down.

'The vast majority of the time, features are waiting in a queue.'

Curt Hibbs, Steve Jewett, Mike Sullivan, The Art Of Lean Software Development

In a system that had no queues at all, the cycle time (the time taken on average to deliver a single set of requirements) would be the sum of all the activities. Queues have a major effect on our cycle time. That means decision gates wouldn't take the week marked out for them in the project plan, or even a day, they'd take the two hours actually spent in discussion. Researching an idea for development (maybe a daring new user interface) wouldn't take four weeks, it would take the actual five days of prototyping potential layouts (four prototyping teams would all run concurrently) plus the actual time to collate and analyse the results – an extra day, perhaps.

Activity 3: Projects are waiting too

Remember your emails from the previous activity? The ones which spend most of their time in your inbox waiting? The same can happen to your projects. We usually only consider the total time a project has taken, which, includes both the active time spent doing the work and waiting time.

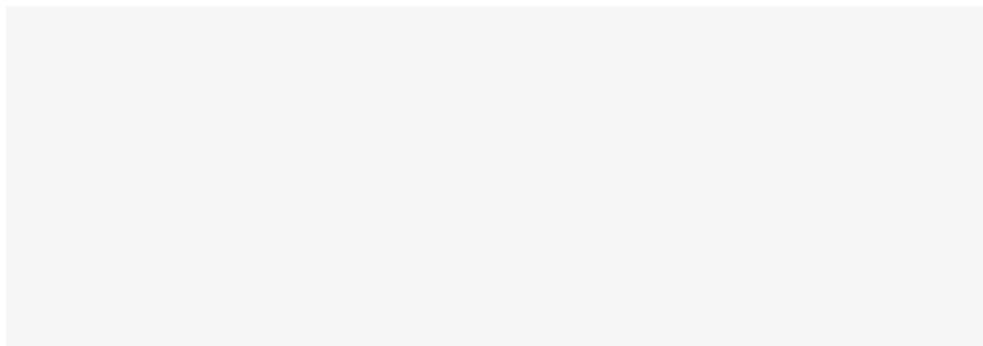
This activity will take 15 minutes and is in two parts.

Pick a recently completed piece of work – this could be a defect which needed fixing, or a change to a live system. Estimate how long the piece of work took to process overall. If necessary, talk to colleagues who worked on it so that you can look at all the activities together.

For the second part of the activity, imagine you could eliminate all the queues within your organisation's development process. Remember queues abound in most development processes: approvals before work can progress; input from a specialist; testing phases, or a deployment window...

If you were to eliminate all such queues, and only count active work, how long would the task take?

Compare the first value you calculated and the second one.



Commentary:

We'd expect that when you compared the two numbers above, there would have been a large disparity.

When we try to achieve faster delivery, we normally focus on how to make the work being done more efficient. The real gains are normally to be made not in work, but in periods of inactivity. Your exercise should help show just how much of an untapped resource for process improvement this waiting time is.

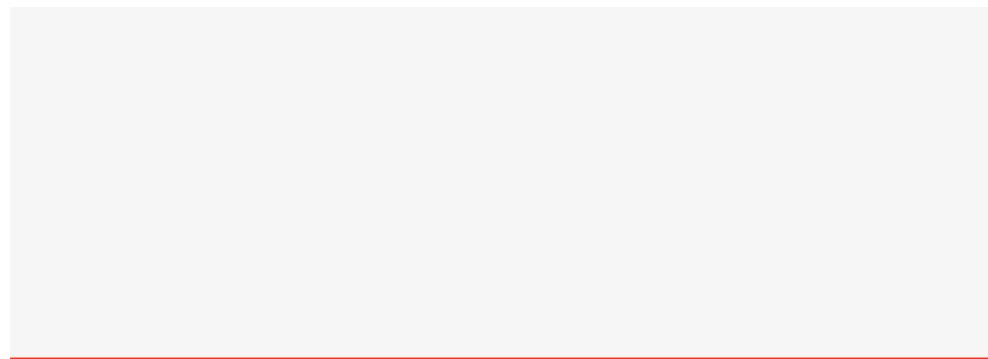
Activity 4: How long will a change take?

In the previous activity you calculated how long an item of work would take if you were to only consider the activities necessary to deliver it. Some development departments pick a change and give it high priority so that it goes through the whole development cycle with a minimum waiting time when its urgent and important in order to test their ability to respond to an emergency.

We know it might be difficult to implement this in your company but it's worth a try. If you decide to go ahead, you will gain valuable insights that you can later use to suggest improvements.

Select one change / item of work and set it up so that it is treated with a high priority. This should allow the item to automatically jump to the head of all queues through the development process.

The time this change takes to travel through your development process is indicative of what it would be with a zero queue length.



1.2. Zero queues

A project run with zero queues takes an extremely short amount of time. It is also, of course, very costly. To have zero queues, you need to keep your people free from other work – your tester needs to be on standby, ready to jump in whenever required; you need a stable of consumers ready to answer questions on your latest idea whenever you want feedback, management must be willing to immediately receive your calls whenever you require his approval on something.

If the fastest cycle time means zero queues, then long queues mean the opposite, and the longer the queue, the slower the cycle time. It takes longer to get to the head of a long queue than a short queue. Valuable work sits waiting for resource to attend to it – this is not only worrying because a slow time to market means you are losing potential sales, but worse, the value in this partially-done work may disappear altogether. The opportunity could pass, a customer might walk away, technology or the environment might change...

In short, queues have a direct economic impact on the business, which means that they have a cost that can be measured. Yet in spite of this, they are rarely tracked or targeted. A company that carefully keeps account of every pound of expense account or hour of overtime is quite likely to be blissfully unaware of the cost of delay to a project caused by long queues.



CASE STUDY: City Airport: The importance of measuring queues

For airports, queues have a direct impact on profitability. The more time passengers spend standing in a queue, the less time they will spend shopping or drinking coffee or any other of the highly profitable activities which airport operators rely on. For City Airport the issue is even more important. Primarily used by business travellers, the Airport specifically tries to cater to their needs with free wi-fi, numerous laptop plug in points and even a free shoeshine. Nice as such touches are, the most crucial issue for business travellers is time – they want to spend more time in the office or at their meetings rather than hanging around in an airport departure lounge – however good the shoeshine is. On its website, London City Airport proudly draws attention to ‘faster check-in times than any other London airport.’

Faster check-in times mean the Airport needs to control queues and controlling queues can't happen without having visibility of flow. The airport needs to know when to add more capacity by opening more desks or security scanners and where to focus staff's efforts. To do this, the Airport needs a sophisticated system and is prepared to update it frequently for the best results.

The airport used to track crowd flow and queues using Bluetooth technology. This relied on passengers having phones switched on, with Bluetooth activated. Only a subset of total passengers did so, meaning the airport was only ever measuring queues via a small sample. The airport upgraded its system in 2012 to use facial recognition software, enabling them to track individual passengers through their entire journey, from check-in desk to boarding. They could look at average queue times between set points and thus predict likely bottlenecks and take measures to avoid them. For example, if queue times were beginning to creep up between check-in and security, they could open another scanner long before a major queue built up which would delay the overall journey. Automation made the job even easier since alerts could inform staff directly whenever times went beyond pre-set limits, without any need for managerial authorisation.

2

WHY DON'T WE SEE QUEUES?

We tend to respond very well to visible queues. We can avoid them (big queue at the bank, I'll come back later); we can get angry with them (complain to the manager and threaten to move your account elsewhere); and we can manage them (invest in automatic paying-in machines to try and reduce queues at the bank).

In manufacturing, where queues are large piles of inventory on the factory floor (and thus present on the balance sheet as unrealised assets), management will expend a lot of effort on reducing them.

The trouble is that the queues that exist in software development tend to be invisible. This is because much of the work is made up of invisible information. Instead of making biscuits, circuit boards, or anything that can be counted easily, the inventory of software development is 'bits on a disk-drive', as an engineering manager from Hewlett-Packard put it. All the ideas and design that we are working on at any one time are our unrealised assets, until we deploy them and turn them into actual value.

Activity 5: Are you too busy? Do you look busy?

Look up from your desk. Look at your colleagues. Can you see how busy they are? How about yourself? Can you honestly tell?

Let's try the following experiment for a day.

Grab a large piece of paper and divide it into three sections labelled To Do; In Progress and Done.

To Do	In Progress	Done

Get a stack of post-it notes.

Think about all the tasks you are already working on. Create a note for each one of them, write a short headline and place it in the In Progress section. Consider also the things you haven't started yet and put them into the To Do section.

As you go through the day, record each new task that arrives creating a new note and recording a headline for it. Move the notes from To Do, to In Progress, to Done as you complete your tasks.

To Do	In Progress	Done
		

By recording your work on notes you were able to see your work more explicitly. Has the increased visibility helped you?

Because our queues are invisible, we find it easy to ignore the effects of them. If we double the number of requirements in a project, there is no warning bell that sounds. Developers in the department might look slightly more stressed, but there is no real way of knowing what the change has done to their work or to how quickly they will complete it. Nor do most companies know what value has been added to or subtracted to the work by this change in input. Now imagine we doubled the number of developers on the project. Everyone would notice that! Not only is there a sudden scuffling for desk-space, but managers would be in crisis meetings trying to find extra budget to pay their wages.

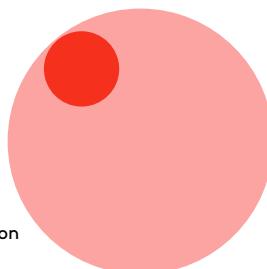
CASE STUDY:

Taurus and the London Stock Exchange



Estimated time: 6-18 months
Estimated cost: £6-18 million

Actual time: 10 years
Estimated cost: £400-800 million



In the elephant's graveyard of IT project failures, the Taurus system for the London Stock Exchange is a mammoth. Estimated to take 6-18 months and with an initial budget of between £6-18 million it eventually took 10 years, costing between £400-800 million – and that's not the best bit ... it still failed.

The analysis of the failures understood that the main issue was caused by trying to accommodate requirements from 280 financial institutions. Because all of these requirements appeared to have value to customers, it was hard for those on the project to see them for what they really were – an enormous queue. Even now, those discussing the project's failure do not truly appreciate the extent of this. The problem was not just a huge scope – the actual activity on even this vast number of requirements and features would not have taken 10 years – it was the inactive waiting time inherent in attempting to elicit, co-ordinate and reconcile conflicting demands from 280 institutions.

The replacement to Taurus – Crest – launched successfully in 3 years. It did not attempt to manage as many stakeholders, recognising that it was more important to keep the initial queue of features and requirements small. Even if this meant ignoring elements of value to customers, it still meant that overall value – getting a working system out there that pleased some people – was more important.

3

WHAT CAUSES QUEUES?

3.1. Work

"Most organisations greatly overload their development resources, creating long queues"

Preston G Smith, Flexible Product Development: Building Agility for Changing Markets

As we increase the amount of work, queues begin to form. Think of opening up your inbox as you arrive at work in the morning and glancing at your in-tray. You have 20 emails to read, plus a little pile of papers in your in-tray. Now, it may be that you are certain you'll be able to complete all of those tasks by the end of the day – nonetheless, at 9am, a queue exists because you cannot work on everything concurrently. If more work arrives during the day – perhaps your boss telephones with a request – your queue will grow.

Even when we are not operating at our maximum capacity, queues still form. This is because the work we do has high variation. Variation is the term used by writers on this subject to mean that our work is unpredictable. As you look through your emails, some of them may require no action at all – you have just been copied in to a mailing list to keep you informed; a few will be asking very simple questions; one or two might require significant amounts of work. Until you open the emails you have no idea whether they contain a task at all, or whether this task will be big or small, easy or difficult. You must have had the experience of cheerfully agreeing to fix a problem, only to discover that it leads you into a labyrinth of connected problems.

It is not only the size of our work that is unpredictable, but when it arrives too. We would say that work arrives in a variable way. When you arrive at your desk in the morning, for example, you might have a heap of emails to answer from someone in a different time zone; after a board meeting a whole heap of tasks might arrive; in August when half the team is on holiday there might be very little to do.

If Fred has been given enough work to be busy 80% of the time, it seems to his manager that he has plenty of free time. As long as you only add a task that takes 20% of his time, Fred should be able to complete it and still have no queues.

While understandable, this is a fallacy.

MR. BUSY

By Roger Hargreaves



Figure 1. Mr Busy – a character from the Mr Men storybooks

Activity 6: The paperclip race

This activity will take 10 minutes. It can be done individually or in small groups.

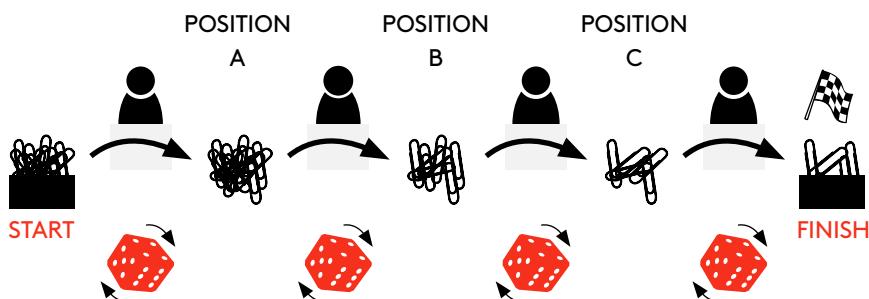
Objective: To pass as many paperclips as possible across the table in 10 rounds.

Preparation:

- A set of paperclips (you may need up to 60).
- A six-sided dice.
- A pen to record progress in the table.

Before you begin, place all paperclips at one end of the table, the start line, with the finish line at the other end, and mark out three positions (A, B and C) in between them. This is the 'racetrack'.

Play:



1. Roll the dice and move the indicated number of paperclips from the start into the first position, A.
2. Roll the dice again. This time move the indicated number of paperclips from A to B. If there aren't enough paperclips in A, move only those that are available.
3. Roll the dice again, passing available paperclips from B to C, and finally from C to the finish.
4. Record the number of paperclips at each position in the table provided to complete the round.
5. Play the game for 10 rounds, repeating steps 1 to 4, remembering to move paperclips across the table only when they are available, and recording the number of paperclips in each position at the end of each round.

	Position A	Position B	Position C	Finish
Round 1				
Round 2				
Round 3				
Round 4				
Round 5				
Round 6				

	Position A	Position B	Position C	Finish
Round 7				
Round 8				
Round 9				
Round 10				

Commentary:

So what happened at the end of the 10 rounds? We'd guess that some paperclips didn't make it, some were delayed, and queues built up. But we couldn't predict where or when. We couldn't anticipate the variability introduced by the rolling of the dice.

Now, we're sorry to reveal that this wasn't actually a race. It was a (admittedly) simplified simulation of a typical workflow. And the variability and unpredictability that we witness in the paperclip race is exactly the same as that which we see in our own work. Even though the odds for moving a certain quantity of paperclips are the same at each roll of the dice, we just can't seem to eliminate queues!

Just like our example of sitting down in the morning to face a queue, because of the way our work arrives, there will be times when Fred already has a queue just dealing with the work that takes up 80% of his time. Fred has a list of 20 things to do following a progress review meeting. This is a queue. Fred should be able to complete them all in 2 weeks, but the work is still queued up for that time. If any one of those tasks takes longer than expected, the queue will get longer. Add a new task – one expected to take 20% of Fred's time – and the queue has just got a lot longer.

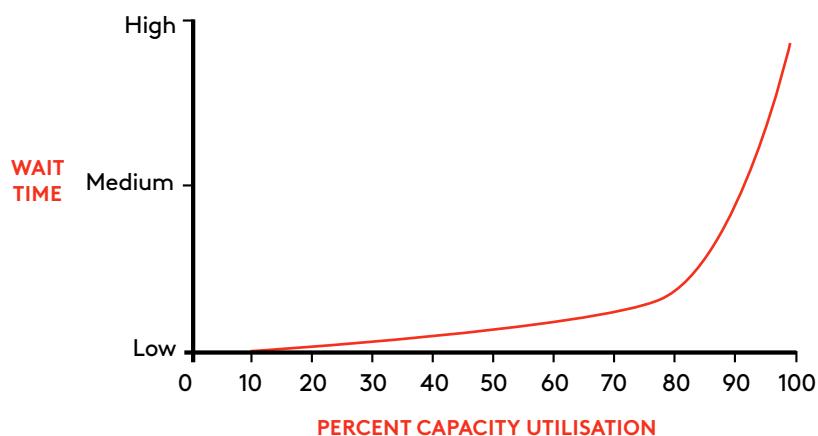


Figure 2. Model showing the relationship between capacity utilisation and wait time

Actually, it would be a big improvement if development departments kept their staff working at only 80% capacity. Most actually claim to work at 98.5%. Indeed, this is seen as a good thing – it shows that the department is ‘efficient’ because everyone is busy all the time. This is especially common in companies where teams or individuals’ time is charged out to clients (consultancies, development and design agencies, etc.). Think of the way lawyers work. A lawyer’s work is typically charged out in 15 minute blocks of ‘billable’ time. Law-firms expect their lawyers to be able to ‘bill’ an astonishingly high proportion of the working day. Lawyers who don’t are seen as less valuable to the firm and eventually may be let go. This mentality aims at the wrong thing - it penalises someone who spends time working on the project rather than filling in timesheets, for example. It also means that as soon as an extra piece of work comes along, or something takes longer than expected to complete, the rest of the work in the queue will be delayed - creating a cost associated with the loss of opportunity that the company fails to set against the cost of having staff less than fully busy.

Earlier we said that in spite of talking about faster development, the actions of many managers suggest that they are more interested in throughput and efficiency. Insisting on full capacity utilisation will lead to longer cycle times and thus must be balanced against cost of delay. As we’ll go on to show, even if efficiency is your primary goal, maximum throughput is rarely achieved by maximum capacity utilisation.

Admittedly, our capacity utilisation is a difficult thing to judge accurately. Work famously expands to fill the time available, so that most of us feel that we work at 100% capacity.

Activity 7: Billable hours

Let’s imagine that your company has moved to a ‘billable’ hours method of working. For a single working day, we want you to consider your time in 15 minute chunks.

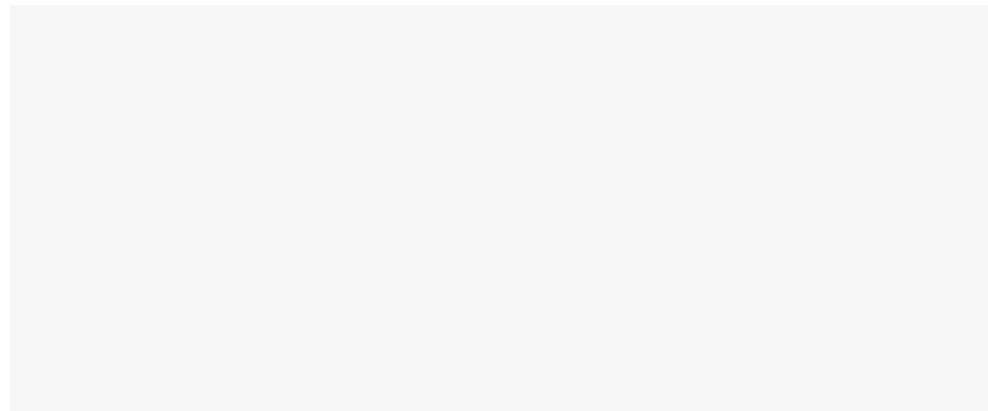
Use a table like this one and assign each 15 minutes to a project as it happens.

Project	Task	Time

For example – when you get in, you probably check all your emails. The first 15 minutes you spend getting an overview of your emails, we will charge as ‘internal’. So in the table you would put ‘internal’ as the project name and ‘email’ as the task recording time of 15 minutes. As soon as you start doing any work on an email or task though – answering a colleague’s query, setting up a meeting etc. – you need to assign this to the relevant project or customer. A meeting about a specific project is assigned to that project, while a chat about what you did at the weekend needs to go down as ‘internal’.

At the end of the day go through your sheet. Normally, companies following this system expect their employees to be able to bill for 7½ hours out of every 8 hours worked (that’s a 93% capacity utilisation).

How many tasks or periods of time did you find it hard to assign to a valuable project? How did the process make you feel? Were you tempted to cheat? How about when you went to get a coffee, or chatted to a colleague about what they’ve been doing? Were you tempted to pretend this hadn’t happened?



Commentary:

This exercise demonstrates two points: first that it is quite difficult to assign all our work to specific projects/customers; secondly is the pressure we feel to appear busy all the time or to look as if you are constantly providing value.

It is an unnatural exercise, which contains its own wasteful buffers. How long did you spend either filling in the timesheet or thinking about it? Did the timesheet really reveal anything about how busy you are or how valuable your work is?

It also makes it clear how very hard it is to do really creative work in such a system. If you want to have an idea about how to improve the website, are you going to put down 30 minutes ‘thinking’ time when no-one has yet given approval for website development as a project?

While few companies in software development actually go all out on ‘billable’ hours, many have an attitude about constant busyness that resembles it. This attitude is what leads to high loading of capacity utilisation, which in turn leads to lengthy queues.

3.2. Long queues

Most of us have experienced a large queue of work. True, we say to ourselves, integrating with that web service has just taken twice as long as predicted, but the next few things might take less time – so it'll all balance out. OK, I just had a big rush of work, but now things will calm down and I can get through my list of things to do (a queue).

The funny thing is we know this kind of optimism doesn't apply in a traffic jam. The more cars arrive, the longer the queue grows. In 1999, Mario the milk float driver took a wrong turn onto the M1 motorway in Britain. Just one slow moving vehicle caused an enormous queue. Drivers had to slow before over-taking him, causing other cars to slow. The knock-on effects of one vehicle moving at 5 mph had to be seen to be believed, while the commentary on Capital Radio from the traffic spotting plane, was definitely one of the highlights of traffic reporting of all time.

As soon as a task takes longer than we expected, a queue begins to form. It is unlikely that this will correct itself through lots of quick and easy tasks arriving in the queue. Instead as the queue gets longer and further out of control, the probability we will be able to get the queue back under control greatly decreases.

The problem is that the rule of thumb in our heads tells us something different. Think of flipping a coin – the chance of getting heads or tails is 50/50. If you flipped a coin and gave yourself a score of +1 for each head and -1 for each tail, you would think that the number of each would be equal and that therefore you would have a mean score of zero. Our score is cumulative, though. You happen to get 4 heads in a row, at this point your next flip has an even chance of taking your score up or down. But you are no longer at the zero point. You are at 4. So you have a 50% chance of going up to 5 and a 50% chance of going down to 3. Each time the trend takes you in one direction, it becomes less likely that you will return to your original starting position of zero.

This is called the 'Diffusion Principle' and there is a neat mathematical proof for it, which we won't include here. It operates in many places, including the stock market. When we buy shares at a certain price and the market begins to go down, we tend to want to hold onto the shares – we don't want to lose money. But as our particular group of shares falls – even if on each day it has an even chance of going up or down – the chance of it returning to the starting point (the price at which we bought the shares), gets smaller the further away we are from the original purchase price. On the stock market, this phenomenon is called a random walk. Stock prices often head in just one direction for long periods of time. Explaining stock price movements after the event, makes us feel like we were in control, even though we weren't.



Figure 3. Milk float holding up traffic



Figure 4. Barclays random walk

The same is true of queues. And sadly the nature of work means that if you've had a long spell of shorter-than-expected tasks, the response is not to say a heartfelt thank you for the good luck and head down the pub, but to load on some more work. Even worse, the Diffusion Principle is predicated on the assumption that the odds are even. In actual fact, experience shows that even with the most honest planning, most of us tend to underestimate how long tasks will take – which suggests that the odds are stacked in favour of tasks taking longer than expected, making the rate at which long queues will get longer even faster.

Queues give us an early warning signal even before the plan changes, since everyone is busy, and completed work is still coming out the other end... it can be hard to spot that something is wrong. But if a queue is forming and this will have a severe effect on the overall cycle time and therefore risk to the project. The queue is actually a much better indicator of future trouble than noticing that an individual piece of work is finished later than expected.

4

WHERE DO WE FIND QUEUES IN IT?

Although every business or organisation is different, there are common areas where you are likely to find queues. Here are three of the key danger areas, with the usual proviso that the extent of the opportunity or challenge is unquestionably individual.

4.1. The fuzzy front-end

We described the fuzzy front-end area of development in the Optimising Flow session, using Reinertsen and Smith's memorable phrase that draws attention to the long periods of inactivity often found in the cycle before the development build actually begins. This is the point when we know an opportunity exists – the idea phase – but have not yet begun all out development.

The queues here may be caused by many factors: approval processes required to initiate funding; a small amount of resource to help with prototyping or research; or even the time required before a full team is free to work on the project. A major reason is that companies invest a great deal of time in ensuring that they only devote resources to the 'right' idea. This causes a big queue at the very start as each idea needs a project plan with cost and expected return on investment (none of which can exist without prior investigation). It is ironic that companies do this in order to minimise their risk, but in the process cause such long delays to their overall cycle time that they actually increase risk of the ideas they do select becoming obsolete.

Activity 8: How long to start building a new product?

Let's imagine that you have a great new idea to build a product in your company.

Think about processes that your company currently has in place; talk to your colleagues and find out what would be required to get the idea approved, and ready for development work to start.

This activity will take 30 minutes.

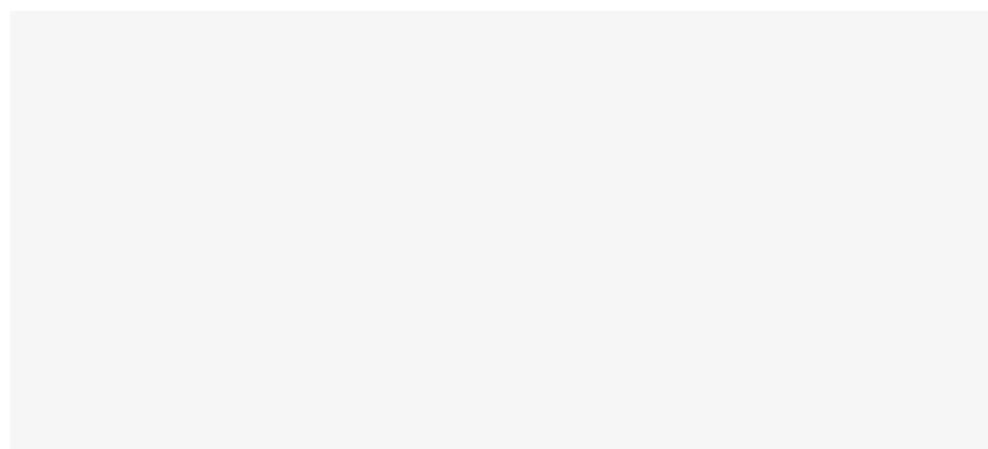
In this activity, we'd like you to list all of the activities, including the sign offs required in order to get your idea to a point that work would begin on it.

As you'll note from the above, these are the activities of the fuzzy front-end phase within your own company.

Go through the list. Estimate how long each item might take or recently took on an actual piece of work. What is a total time required in your company to start building a new product? How might you go about condensing these timescales?

Discuss these findings with somebody who has the ability to make changes within the process – if that's not you.

As a follow up, assuming you have the authority to do so, we'd like you to implement one of the changes you have identified.



CASE STUDY:**Airport expansion - London vs Beijing**

Figure 5. Beijing airport

When Beijing won the Olympic bid in 2001, one thing was very clear – it would need to expand the airport. Terminal 1 and 2 were already operating at capacity and air traffic was growing at 20% a year. Construction began on the new airport in 2004. Ten thousand villagers had to be relocated from the area surrounding the site, while 50,000 workers were brought in to complete the site in the proscribed three and a half years. Half a million tonnes of steel and two million tonnes of concrete were used in the building, which extends for almost three kilometres and is designed to be Feng Shui compliant. It is six times larger than Heathrow's T5. There are 300 check-in counters, 451 elevators and a system of baggage carriers able to move up to 20,000 bags per hour over 60 kilometres of track at a speed of 7 metres per second. Trial operations began in February 2008, when the first airlines moved in to ensure the new terminal was ready for full and perfect operation at the beginning of the Olympic games. It is currently the second busiest airport in the world and plans are already underway to open an entirely new airport for Beijing in 2015.

When London won the Olympic bid in 2005, airport officials and civil servants must have groaned. Plans for a new London airport have been mired in controversy for decades. London Heathrow is the third busiest airport in the world and plans for a third runway, first proposed in 2003, were given the go-ahead in 2009 and scrapped in 2010. Plans to build an entirely new 'transport hub' for London in the Thames Estuary have been mooted by the London Mayor, who has led a scoping and feasibility study since 2008. Given the need to accommodate more passengers than London airports can currently handle, and the very real cost to London and the UK as travellers bypass it in favour of new routes based on Dubai or Paris instead as a hub, the need to make provision is agreed on by almost everyone. Each proposal faces significant protests from those who live near the proposed sites. But as the debate rages on, and no decision is taken, the problem and the cost of delay only gets worse. In fact the first plan for an airport in the Thames estuary was made in 1943. That's a 69 year delay. Admittedly the need to build consensus may be more important in a democracy, but there are few better examples of a fuzzy front-end made worse by political wrangling that won't be resolved until the cost of delay has reached critical proportions. And we bet any new Thames Estuary airport won't be built according the principles of Feng Shui either.

4.2. Specialisms

When your doctor refers you to a specialist it may take two weeks to gain an appointment. At the hospital itself you probably have to wait for up to an hour for your appointment that itself only lasts a few minutes. This is because the specialist is a scarce resource whereas your time (according to the hospital's point of view) is not.

Specialisms in IT work in exactly the same way. Just like the hospital, we tend to manage them for maximum efficiency – keeping the specialist fully utilised. As we know, this is the recipe for a queue. Employing more specialists is expensive. Getting your specialist to train others is difficult – since specialists are already busy, finding time to train others is a significant investment that few companies make. Also the specialist herself may not co-operate since her status and pay probably depends upon the rarity value of her specialism.

Projects can have dependencies on external individuals or external teams. You may require a specialist with a particular skill (architect, expert in SAP, etc.), or the product of a group of specialists' skill, (for example, a specific component or piece of hardware).

Activity 9: Generalising specialists?

You can experience the effect of employing specialists by playing the following game. Ask at least two colleagues to help you.

This exercise should take you no more than 20 minutes.

Objective: Create as many paper aeroplanes as possible in six minutes.

Specialist: A member of the team with experience in a specific skill.

Preparation:

- A time keeping device.
- Paper for the aeroplanes.
- Pens, pencils or crayons to paint the planes.

Play:

1. Set-up a production line to make paper planes.
2. Select one person from the team to be a specialist painter.
3. In 6 minutes try to make as many planes as possible:

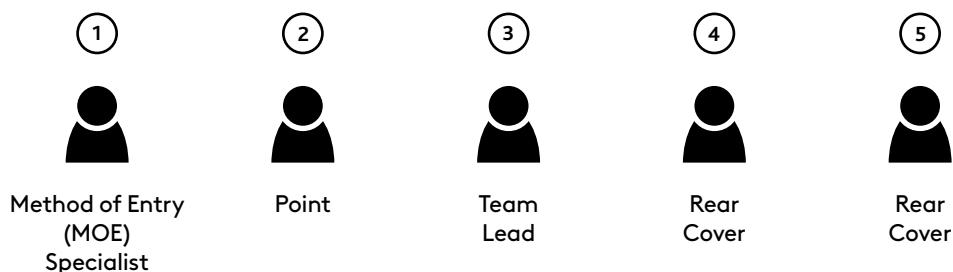
- a. Non-specialists on the team fold paper planes
 - b. Pass it over to the Specialist
 - c. The Specialist should adorn each plane with six windows on each side and an airline logo.
4. Count the number of finished planes at the end, discount any which are missing artwork.
 5. Repeat the process spreading the work of the Specialist to the other members of the team.

**Commentary:**

As you will have expected, the first round saw a bottleneck quickly forming at the Specialist, whereas the second didn't. Typically throughput would be seen to increase in the second model.

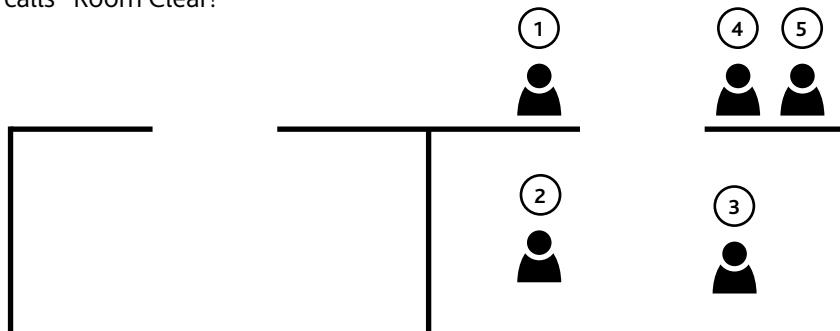
CASE STUDY: Generalised specialists in close-quarters battle

Most of us are familiar with the scene from action films. An army (or police) unit approaches a door. A small team of five approaches a door behind which lurk enemies or hostages. The team stand on either side of the door. The numbers refer to the numbers on the diagram below. 1 is Method of Entry (MOE) Specialist, 2 is Point. Behind Point, 3 is Team Lead, 4 and 5 are Rear Cover.

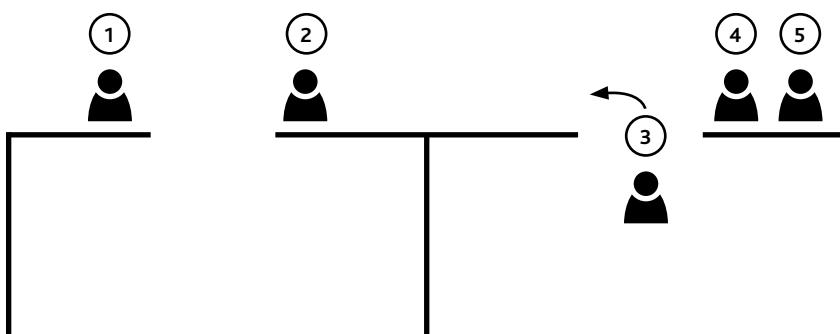


The Team Lead calls "strike". The MOE Specialist breaks down the door with a well-aimed kick. The Team Lead throws a flash-bang grenade and Point immediately enters the room, followed by Team Lead. Rear Cover move up in case they are required to enter.

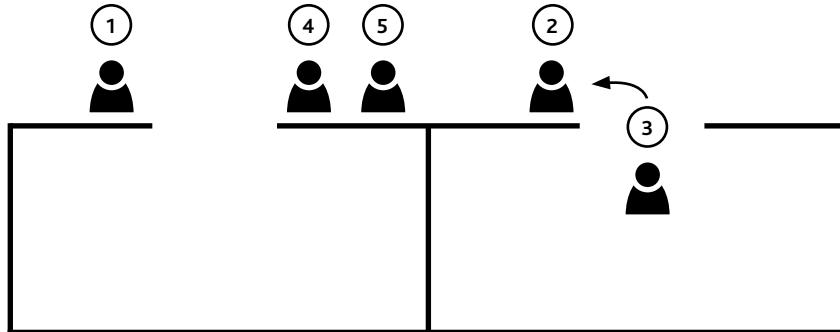
Point calls "Room Clear!"



In a fixed role situation, Point and Team Lead need to exit the room and move on to the next doorway, where the MOE Specialist will be waiting, and Rear Cover will shuffle in behind. This takes up valuable seconds, giving the enemies time to kill the hostages, escape or throw their drugs down the toilet (take your pick of scenario!).



In a ‘generalising specialist’ method, the MOE Specialist can move on to the next door way, with Rear Cover running up to take positions as Point and Team Lead, while the previous Point and Team Lead become Rear Cover.



Not only does this make the whole team faster, but it means that if one of the team members is injured or killed, the others can take his place without the team’s effectiveness being desperately compromised.

The analogy compares directly to software development. A truly cross-functional team of generalising specialists can work much better on concurrent processing which reduces wasted time by ensuring a continuous flow (and increasing throughput as well). It doesn’t mean specialists are unnecessary – the MOE Specialist may still be required to tackle the hardest barriers or at least advise – but on a ‘normal’ barrier where anyone can wield the forced entry battering ram, for example, then any team member can take on the role.

4.3. Environments

Big queues in software are not always about people. They are quite as likely to be caused by a different resource: hardware and environments.

Hardware is frequently a constrained resource, either in itself or because of how it is set up. Teams often need to share a test server, and therefore access it by scheduling in time slots. This makes complete sense to those considering efficiency: teams only need the full capacity of a server some of the time, so they should share a resource rather than having a dedicated one.

Sometimes the issue is that the department responsible for controlling hardware, infrastructure and environments is more interested in efficiency and utilisation of the servers than about expediting development flow. At other times the problem is more likely that the development team themselves have failed to recognise that the hardware or environment can become a bottleneck – by not setting up or preparing in advance they slow the process and form a larger queue.

TOTAL COST = COST OF CAPACITY + COST OF DELAY

As we know total cost depends upon a combination of the costs of the capacity (in this case the infrastructure) and the cost of delay to the project. If the project cannot proceed because it needs to wait for its ‘turn’ on the testing server, then this delay cost should be factored in to any decisions about infrastructure. Nor is that the only cost. Because testing may require a different configuration, there is also a significant switching cost to factor in every time we use environments for different projects. Failing to prepare is a lack of investment by the team in their own project and reveals a lack of awareness of the need to optimise flow.

Since most teams prefer to blame queues on something external (we can’t get a session on the test server until next week) rather than on themselves (we should have run that test in half an hour rather than spending all day fiddling about with the set up), the focus in recent times has been to find a technical solution through virtualisation technology. Sadly, the benefits of this have yet to really deliver. In any case the element which is about flow prior to a bottleneck – teams not preparing set-up of instances, switching costs etc – remains a major source of delay.

Capacity in hardware and the environment is one of the hardest areas to optimise. Infrastructure requires budget approval (with its own high delays in the fuzzy front-end), it requires capital outlay (often difficult when linked to a single project), lengthy periods of time to set up (more delay), and it often requires many specialisms to make sure environments are set up in the best possible manner (and we know about specialists and their queues). Changing the way teams manage their time and processes ought, in theory, to be simpler. It is something over which we have control and therefore lends itself to the optimisation processes we have discussed.

5

CONTROLLING QUEUES

There is an exciting conclusion to the problems caused by queues: by controlling them, we can shorten cycle time.

This feels slightly counter-intuitive to many organisations, who are used to announcing a date by which all work must be finished, or at any rate concentrating on how the work can be done faster. It is something of a shock to ignore those measures and instead concentrate on reducing the queue – on the inactivity that is part of every project, rather than the activity. To understand this in a concrete sense, think of the supermarket opening extra tills to enable customers to flow through faster.



Figure 6. People queuing in a supermarket

Of course, what the supermarket is doing is adding temporary capacity at moments of high pressure to make sure we capture the customer sale. This is a relatively expensive solution, although it tends to be the one we think of most immediately. (Actually, unlike the supermarket we tend to think of adding permanent capacity, which is even more expensive). Indeed, the combination of our awareness of the expense of capacity mingled with our lack of understanding of the real cost of queues, is the main reason why we tolerate queues and suffer the associated lengthy cycle time.

All is not lost! There are other methods of controlling the queue which can reduce it and manage it, without increasing capacity – several of these are cheap and reversible if they do not work.

5.1. Making the queue visible

Gaining visibility of the queue is the first and most important step in its management. We'll go into this more in the Work In Progress session, but at its simplest level we can simply measure how long work takes to pass through the queue. A more complex but potentially helpful visual depiction of the queue is the Cumulative Flow Diagram (CFD) – this tells you not only the size of the queue, but whether it is exacerbated by a large number of arrivals or a lengthy service time that results in few departures. It is particularly helpful for spotting emerging queues. For help in creating a CFD, go to the Technique Library.

5.2. Estimating queues

"Little's Law will rule the world."

Corey Ladas, Scrumban

John Little formulated a useful equation, Little's Law, to use when considering queues. It equates average waiting time with queue size and processing rate. It is very robust, applicable to everything from the overall system to the individual product queue and it is also simple to explain compared to other queuing theory concepts.

$$\text{AVERAGE WAITING TIME} = \frac{\text{QUEUE SIZE}}{\text{AVERAGE PROCESSING RATE}}$$

Imagine a jar containing a thousand lollipops (the queue). A hungry child can remove one lollipop a second (the processing rate), meaning it will take nearly 17 minutes to empty the jar (the queue time).

Now we can calculate the total wait time to clear all the items in our queue. Even better, perhaps, we can pick any item in the queue and because we know where it is, we can calculate how long until we will have to wait get to it. This works even when items have high variation – some long, some short – because our processing rate is an average based on a mixture of long and short tasks.

If you telephone a customer service line and are kept on hold, you will often hear a voice telling you, "you are 4th in the queue". While this is intended to encourage you, it is not actually very helpful. A more effective way of keeping people waiting, (or encouraging them to ring off and call back at a quieter time), might be to use Little's Law in order to tell them, "your call should be answered in approximately 8 minutes time".

It should be easy, and yet we don't know of many companies who are doing it, apart from one which uses a less dynamic, but still effective method: Disney.

If you have taken a small child (or a big child, or didn't even bother with a child at all...) to Disneyland or Disneyworld, you'll probably have stood in a queue for one of the rides. These have regular signs telling you: '45 minute wait from this point' reducing in 5 minute steps until you reach the head of the line. Average wait times for the key attractions may even be posted at the entrance to the park.

Some companies have stopped estimating a delivery date for a project altogether. Instead they use the queue and processing time to provide the answer to how long any given project will take. Huitale, a Finnish software development agency, use an average wait time to announce when each feature will be ready. If this is too slow for the Product Owner, it is up to him to reprioritise and move the feature up the queue. Interestingly they call their wait time, 'Disneyland wait time' since it is approximate.

Of course, Disney lines move at a regular speed (20 onto the ride at a time), but they have also added some variation. The 'Freepass' allows people to jump to the head of the queue. Some of these are given out to guests at the hotel or they can be bought, or any visitor can collect one from a special booth.

This is reminiscent of something that happens in development. Just as the Disney ride queue may have visitors brandishing freepasses arriving, so the team may have new projects, features or tasks arriving. Some of these may be more important than what was already in the queue, which means that our wait time needs to be constantly updated. Our queue may not progress in an orderly fashion because we are prioritising, or sequencing, tasks from the queue according to decision rules either that we impose ourselves or that we are guided to by a customer, Product Owner or manager.

Activity 10: What is your email Disneyland wait time?

Now it's your turn to do some calculations.

This activity will take 10 minutes.

Open your inbox and count all the emails that require some action on your behalf. This is your queue size.

Look at your sent items and count all the emails you have sent in response to an inbox item in the past seven days. Divide the number by seven. This is your processing rate in emails per day.

Now divide the number of emails that need an action in your inbox (queue size) by the obtained processing rate.

Emails requiring action	Emails sent in response in the last 7 days	Processing rate in emails per day	Wait time

Commentary:

This is your wait time, in days. This is how long, on average, your colleagues will have to wait for a response when they send you a new message.

Note that the wait time you have just calculated is based on the assumption that your current processing of emails is constant (for example, that you deal with your emails as they arrive). We have also assumed for the practical purposes of this activity that you respond to all actions with an email. In practice, we often apply some arbitrary order. This is discussed in the next activity.

5.3. Sequencing

Sequencing the queue depends upon our knowing several pieces of information about the project or task: value, urgency and duration. These do not have to be actual figures (although that can be helpful), they might simply be labels instead. Value – high; urgency – very; duration – short.

The priority order depends upon a ratio between these figures. Imagine a journalist with three articles to write: one is for the local paper and he has nearly finished it; the second is for a major national paper but he has not yet begun the research, and the third is for a friend's website which he promised to write two weeks ago and hasn't yet delivered.

Obviously, the second article is the most valuable both in terms of kudos and money (highest value), the third is the most overdue (urgent), and the first is the nearest to being finished (shortest duration).

What should he do?

Although the second project has a greater potential value, the journalist only needs to invest a tiny amount of work –in order to release the value for the first article. If he hadn't begun either article, the national paper would have taken priority. But because he has already invested time in the first project, it is worth finishing it to release the value.

But what about the overdue article for the friend's website? That's the late project – shouldn't he give that priority? A friend's website has a low cost of delay. That means it gets sequenced behind a project with a HIGH cost of delay, even one on schedule.

There's no need for the journalist to get out a calculator or assign figures. The mental arithmetic to work out the sequence is not hard:



Sometimes decisions go down to the wire, and at that point it can be helpful to have an actual calculation to help guide you. Generally, however, you can rely on a rougher, mental calculation: we begin with tasks for projects with the highest value. Where the value is equal (or two tasks are from the same project) the shorter task should take priority. Firstly, the shorter task delays or blocks the resource for a shorter time and it may be that a new piece of work will arrive with a higher priority. More importantly, it may be that by completing the shorter task we can realise its value.

Which can be summarised as two new rules of thumb:

**HIGH VALUE BEFORE LOW VALUE
SHORT JOBS BEFORE LONG JOBS**

Activity 11: Can sequencing help your busy inbox?

As you have read in this section one of simple ways to eliminate impact of queues is by sequencing their tasks. You can do this using the given two rules of thumb.

Open your inbox and select 10 emails you have to deal with.

Pick 10 index cards and write a short description for each of the emails. Go through the cards and estimate the value for each of them; it's sufficient if you only indicate Low, Medium and High. As you think of the work required, also indicate whether you think the task will be Short, Average or Long. Write one of the following categories on the card:

- High - Short
- Medium - Short
- High - Average
- Low - Short
- Medium - Average
- High - Long
- Low - Average
- Medium - Long
- Low - Long



Now order your cards. Do this by putting the cards in the order indicated by the list above. The order in the list was created by combining the two rules of thumb.

Deal with the identified emails in the established order.

How did it go? Did it make a difference to your usual sequencing? Maybe look back at your calculations from Activity 11 to get a better feel for any change.

Commentary:

We are not suggesting you start writing out cards in order to manage your emails. In fact, if you need to do this then we recommend employing a secretary. But, you may find this approach helpful at a more macro level – managing tasks within projects or deciding priorities between projects themselves.

5.4. Identifying the problem area: bottlenecks

"Any system can produce only as much as its critically constrained resource, were it not so the output of that system would tend to infinity or zero"

Eliyahu Goldratt, Theory of Constraints

We have discussed that queues cause delays, and that long queues get longer. It follows that the longest queues cause the most damage – indeed, a disproportionate quantity of damage. This is because delay is cumulative....

Let's imagine three projects in our queue, creatively entitled Project A, Project B and Project C.

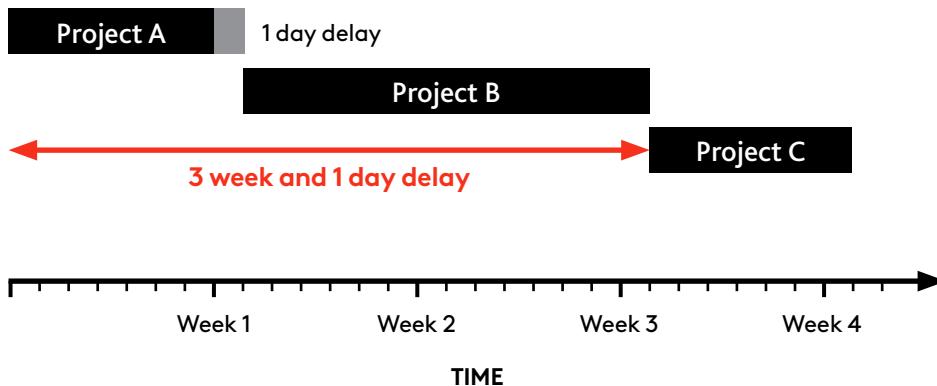
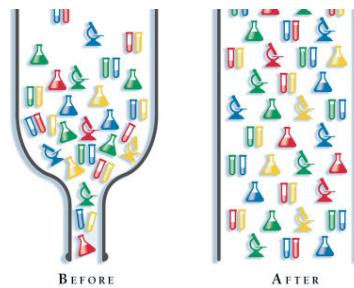


Figure 7. Three projects in a queue and the effects of a delay

Even though it is only Project A that suffers a delay of one day, we can see that it affects all the other projects. Project C's start is delayed by 3 weeks and 1 day, rising from 3 weeks. If we made this delay a cost (it doesn't even have to be a cost of opportunity, perhaps we are paying a fine for delivering late), then you can see that every project has to pay the fine, because each of them has suffered a delay – even though only one of them overran, assuming that we cannot start a project until the other ends.

Queues tend to form around any single process within the system which for some reason has a limited capacity and which therefore constrains the capacity of the entire system. We often call this a bottleneck. As soon as a queue forms here increased cycle time will result.

Bottlenecks can form at any number of points, wherever there is a potential mismatch between capacity and demand – and remember the queue can begin even before we have reached full capacity. Specialists are a common bottleneck, because a scarce resource (often an expensive one) tends to be managed for efficiency (the specialist must be at full capacity). Decision points, both internal and external, frequently become bottlenecks as does any hand-off point between departments or areas.

Most of you will have come across Gantt or PERT charts which are employed in project management to determine the critical path on a project. Bottlenecks that form around any process on the critical path are obviously directly creating delay, but those off the critical path should not be seen as ‘free’ queues. After all, activities here will potentially find their way on to the critical path eventually. Remember our example about driving your anxious Mum to the airport in our Optimising Flow session? We mentioned non-critical activities such as filling up with petrol. With half a tank, the activity is non-critical, but it will become critical at some point – possibly at a point where you have to delay something more important to service it.

A quick word to say that although critical path analysis can be very useful as it focuses on the time line of delay, a drawback in the systems mentioned above is that they have a precedence approach. This assumes that each task must be completed before the next can be begun. It is an assumption that can increase queues through large batches (which is covered in the Batch Size Matters session).

Before the bottleneck

When a queue forms at a bottleneck, we want to intervene quickly to control it. However, before we go on to discuss dealing with the bottleneck itself, we should also examine the processes leading up to the bottleneck. Take the x-ray scanner at an airport. To break the bottleneck you might need to add another scanner – an extremely costly business. However, by paying attention to the process just before the bottleneck, it can be made easier. Signs along the queue remind passengers to take off their shoes, remove their belts, take out any coins, etc. in advance. Staff help direct and prepare passengers – a classic example of the benefits to be gained from looking at overall flow rather than a process in isolation.

The same principles can be applied to software development. Take two of the common bottlenecks listed above. By preparing work for specialists, we can make their job as easy as possible to help minimise the queue. Decision makers rely upon the quality of data offered to them – insufficient data will exacerbate the bottleneck. Reinertsen commented that many development teams complained to him that their senior management took too long to make decisions. He asked to see the paperwork the teams had submitted. In every case, Reinertsen felt that they had failed to clearly state the benefit and expected cost of their project, making it more likely that a decision would require fresh information and therefore be slow.

6

HOW DO WE QUANTIFY A QUEUE?

Queues cause delay.

Delay costs money.

Removing queues costs money.

It sounds like a kind of logic puzzle, but it is actually impossible to resolve satisfactorily unless we take steps to quantify the true cost of delay. Remember that some projects are more vulnerable in the face of delay than others. Every project will have a different cost of delay, and therefore every company must make a slightly different decision about how much delay can be tolerated, how long their queues can be allowed to grow before energy or money is invested in reducing them.

Tiffany & Co, the famous jewellers, has a very high level of staff on its gold desk: the risk of annoying a customer who might be about to spend thousands of pounds justifies keeping staff at low utilisation. On the silver desk, fewer staff serve more customers, the lower value of each sale means that making some customers wait is an acceptable trade-off. Although queues will always cause delay, delay may be worth suffering if throughput or efficiency are considered more important, as is true at airport security, for example, or in the average GP surgery in Britain.



Figure 8. Staff at their desks in a Tiffany store

Of course, while sometimes businesses can operate on a best guess (my gold-buying customers mind waiting, my silver ones don't), for our decisions to be meaningful, we need to do two things: measure the length of our queues, and calculate the cost of delay for a project.

Activity 12: What are your classes of service?

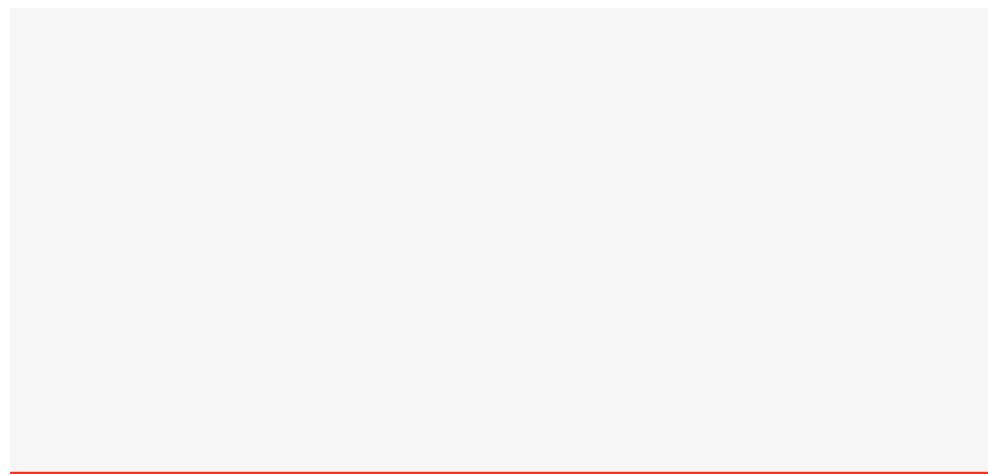
You've just read about how Tiffany & Co treat the customers at their gold and silver desks differently. They appreciate that there is a difference in value between the two types of demand and set themselves up to service that demand in different ways.

This activity will take 15 minutes.

Consider the way your organisation delivers services to its customers. Is all the demand of equal value?

Can you identify categories of demand that differ? How would you service the queues for each of those types of demand?

Consider what additional costs might be associated with processing the work in some of the listed categories. Is the additional expense worth it?



6.1. When do we do this?

At the macro level, when launching a project, you will almost certainly have to complete a business case which projects the sales you expect at launch, takes into account competition and the expected life cycle of the product. You will be asked to estimate a cost of capacity required by the project. At this point, doing the cost of delay calculation ought to be a standard inclusion. Since you have all the assumptions in front of you – it really shouldn't be too painful! For help in doing this, consider the content in the Prioritisation session.

Don Reinertsen in his excellent book *The Principles of Product Development Flow*, goes into the precise calculations for how and why you should keep this updated and drills down to a lower level, not just the project as a whole, but features within it. He then uses this cost of delay to help sequence the queue.

We would argue that granularity in these calculations is often unhelpful. The higher the overall cost of delay to the project, the more value there is in drilling down further. In a project with a very high cost of delay, for example, we might want to be sure that each feature in the backlog had a cost of delay calculated. The narrower the margins at which you are working, then the more precise you need to be in your assumptions and numbers. The problem comes if you are doing this so much that you end up employing an accountant (thus increasing your cost of capacity) just to keep up with the maths.

Since many of your numbers will be estimates in any case, we can use far simpler methods: prioritising high cost of delay projects over medium and low; and prioritising features that add the most value to the customer within the project. This remains only a general rule of thumb – there may be occasions in which a project with low cost of delay is important for other reasons. Consider compliance with regulations. The work needed to ensure this is done (cost of capacity) may often be far higher than regulatory fines, but the fact is that as a responsible organisation you are likely to put good citizenship and corporate reputation ahead of an economic benefit.

By making flow visible through a Cumulative Flow Diagram or a simple list, we can see a long queue state emerging. At this point we may need to quantify its true cost in order to help us make difficult choices. After all, we know that long queues cause the most damage so in order to stop this queue spiralling out of control and wreaking economic havoc in our business, we need to tackle it swiftly.

For example:

A one year project has a cost of delay of £5 million per month. About half way through the project we notice that a queue is forming. If we continue processing the work at the current rate we will be three months behind schedule. By employing three new temporary staff at a cost of £500,000 per month, we think we can get back on track.

A projected cost of £15 million in delay is set against a known cost of £3 million. All other things being equal, we should employ the staff.

7

CONCLUSION

In this session we have investigated in detail why and how queues lead to longer cycle time and project delays. We hope it is also clear that while queues cause delay, this does not mean they are in some way ‘evil’ in and of themselves. A zero queue level will ensure the greatest speed, but the cost may be very high – if this high cost is not acceptable to the customer because for some reason they do not value speed, then zero queues could turn out to be disastrous for the company.

The careful balancing of various different elements from cost to speed in order to achieve the desired outcome is exactly what good management is about. What is unacceptable is to be unaware of whether you have any queues and what their significance and impact might be. Measuring them in some way – whether by estimation or in detail – allows you to take control and then decide on the right action to achieve your business objective.

Learning outcomes

Now that you have completed this session, you will have an appreciation of:

What a queue is

- Wait time that any given task or project spends in the system as opposed to activity time
- Queues increase cycle time
- Conversely zero queues ensure the fastest possible cycle time

Why queues are invisible

- In IT inventory consists of ‘information’, rather than physical objects
- Queues are rarely measured
- What causes queues
- A lack of spare capacity will ensure a queue at some point because of the high variation of IT and all knowledge-based work
- The cumulative nature of delay ensures that long queues get longer, cause disproportionate damage and easily spiral out of control

Where queues are found in IT and software development

- The fuzzy front-end – project approval and funding prior to development
- Specialisms – any specialism has the opportunity to become a bottleneck, but rigid roles or a lack of real cross-functionality also has an impact
- Environment and hardware – often ignored in IT, a physical resource can also cause delays

Methods for controlling queues

- By making the queue visible we are able to measure it
- Methods for estimating how many items in the queue, how long it will take us on average to get to a given item in the queue and how long an average task will take to cycle through the queue
- Sequencing items in a queue by value and size allows us to minimise the economic damage from a queue
- Identifying bottlenecks within the flow enables us to make choices about widening the bottleneck or preparing work for it more effectively

How to quantify a queue

- Assigning a monetary value to cost of delay and hence to the queue enables us to make difficult choices about managing the queue
- Since an exact calculation can be time consuming and is based on assumptions, consider when an approximation is good enough, and when a more precise figure is required

When action on queues is necessary

- Since long queues get longer and cause more damage, prompt intervention is crucial

BIBLIOGRAPHY

- Future Travel Experience**, 2012. London City improves passenger queue measurement with facial recognition technology. Future Travel Experience. [blog] 1 March, Available at: <<http://www.futuretravelexperience.com/2012/03/london-city-improves-passenger-queue-measurement-with-facial-recognition-technology/>>. [Accessed 10 October 2012].
- Goldratt, E., Cox, J.**, 2004. The Goal: A Process Of Ongoing Improvement. 3rd Revised Edition. Gower Publishing Ltd.
- Hibbs, C., Jewett, S., Sullivan, M.**, 2009. The Art Of Lean Software Development: A Practical And Incremental Approach. O'Reilly Media.
- Human Recognition Systems**. Gatwick Airport: Deploying MFlow Journey in the new South Terminal Passenger Search Area. [online] Available at: <<http://www.hrsid.com/clients/mflowjourney/32-Gatwick-Airport-Deploying-MFlow-Journey-in-the-new-South-Terminal-Passenger-Search-Area>>. [Accessed 10 October 2012].
- Human Recognition Systems**. London City Airport: Measuring passenger flow with MFlow Journey. [online] Available at: <<http://www.hrsid.com/clients/mflowjourney/33-London-City-Airport-Measuring-passenger-flow-with-MFlow-Journey>>. [Accessed 10 October 2012].
- Ladas, C.**, 2009. Scrumban - Essays On Kanban Systems For Lean Software Development. Modus Cooperandi Press.
- Larman, C., Vodde, B.**, 2008. Scaling Lean & Agile Development: Thinking And Organisational Tools For Large Scale Scrum. Addison-Wesley.
- Project Case Studies**, 2011. Project Failure - Taurus. [online] Available at: <<http://www.projectcasestudies.com/?p=88>>. [Accessed 3 August 2012].
- Reinertsen, D.**, 2009. The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas.
- Salesforce.com**, 2007. Large Scale Agile Transformation: How Salesforce.com Revolutionized Their R&D Development Methodology In A Big Bang Way. [online] Available at: <<http://www.slideshare.net/sgreen/salesforcecom-agile-transformation-agile-2007-conference>>. [Accessed 23 January 2012].
- Smith, P.**, 2007. Flexible Product Development: Building Agility for Changing Markets. Jossey Bass.
- Smith, P., Reinertsen, D.**, 1998. Developing Products In Half The Time: New Rules, New Tools. John Wiley & Sons.
- Taipale, M.**, 2010. The Huitale Way - Our Value Stream Map. The Huitale Blog, [blog] 23 March. Available at: <<http://huitale.blogspot.co.uk/2010/03/huitale-way-our-value-stream-map.html>>. [Accessed 3 August 2012].



valueflowquality.com

emergn.com