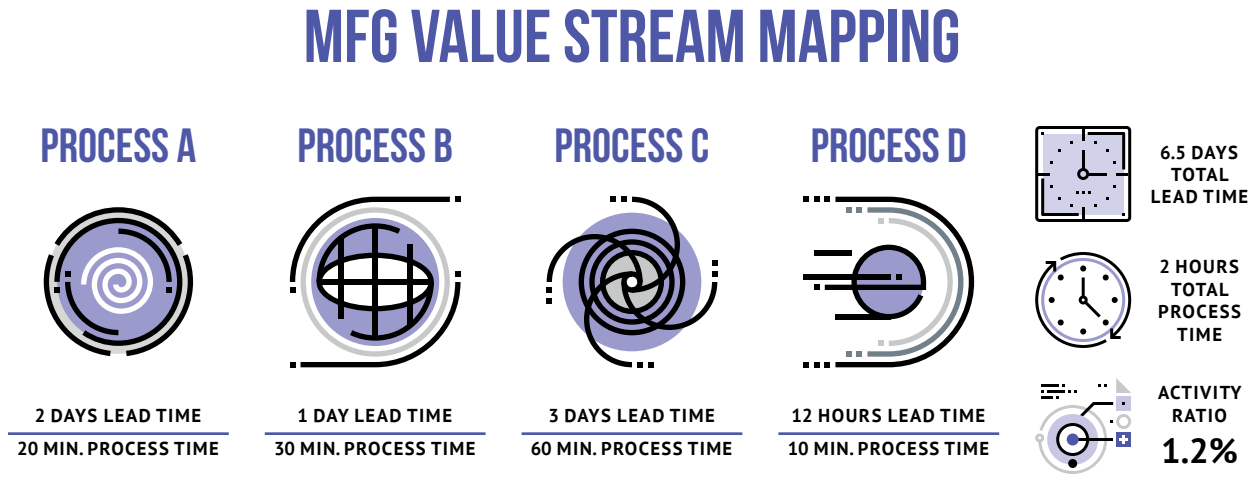
Techstrong Research

COMMON MISTAKES IN APPLYING VALUE STREAMING MAPPING TO SOFTWARE

A WHITEPAPER BY GARY GRUVER

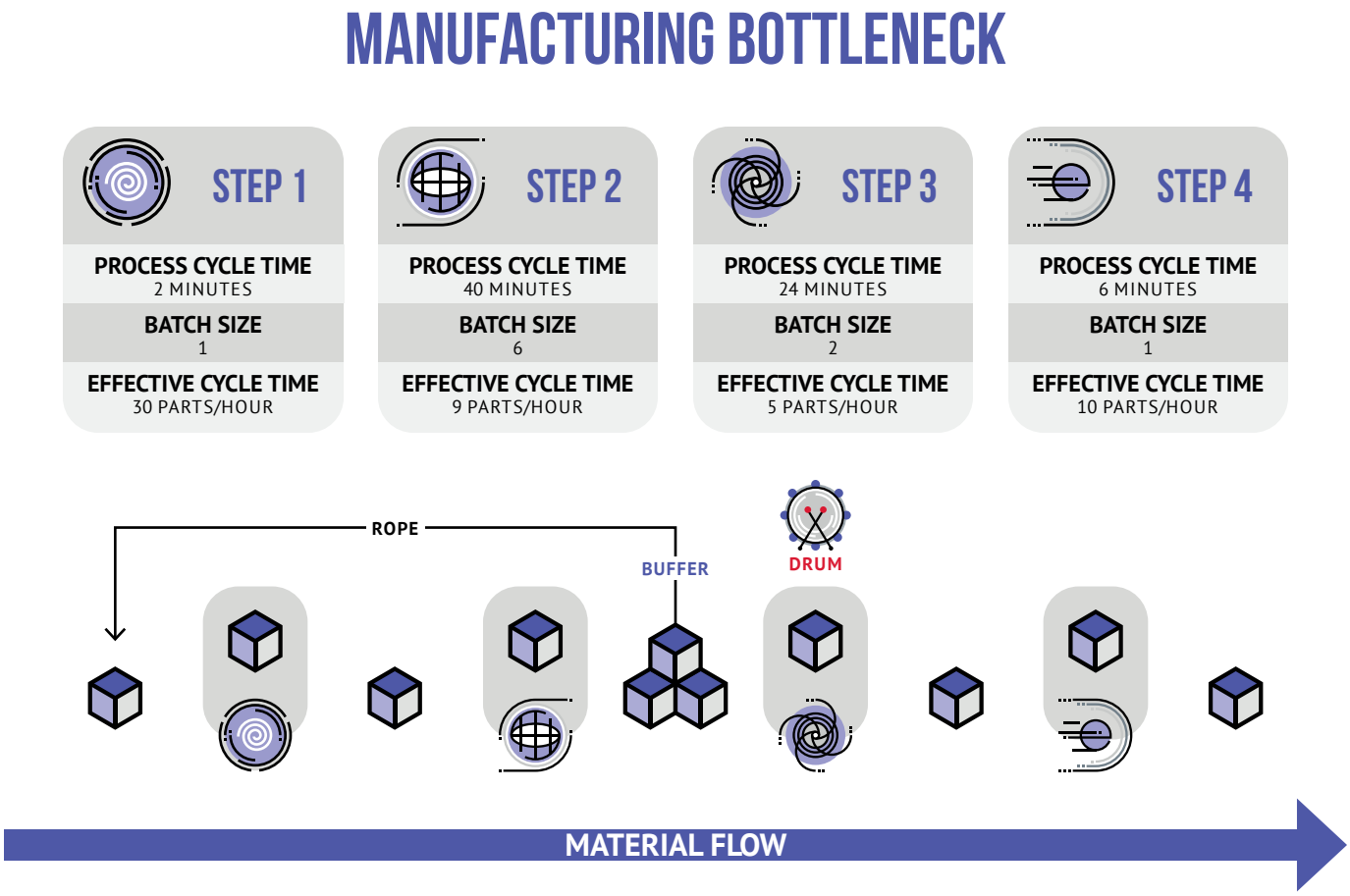
VALUE STREAM MAPPING is starting to be used more and more in software organizations as a tool for identifying opportunities for improvement. It is an approach used to make waste and inefficiencies visible that started in manufacturing and has since been used for a variety of applications including software. Recently as I have been consulting with different organizations, I have had the opportunity to see lots of different maps they have created that I think highlight common mistakes. Or more accurately stated how when applying the classic approach of Value Stream Mapping to software it frequently doesn't do a very good job of highlighting the biggest sources of waste that are slowing down organizations. My goal in writing this white paper is to make these issues visible so everyone can be more efficient in identifying opportunities for improvement and transforming their software development and delivery processes.

Value Stream Mapping is tool that was developed in manufacturing to identify opportunities for improvement. It starts by mapping all the process steps required to create a product or value for the customer. Once you have each process step identified then you need to collect a couple of metrics for each step. The first is the lead-time or the total time the part spends in that process. The second is process time or how much time is required to add value in that step. For the total value stream you can then calculate the activity ratio which is total process time / total lead time to identify the magnitude of the opportunities for improvement as shown in the graphic below.

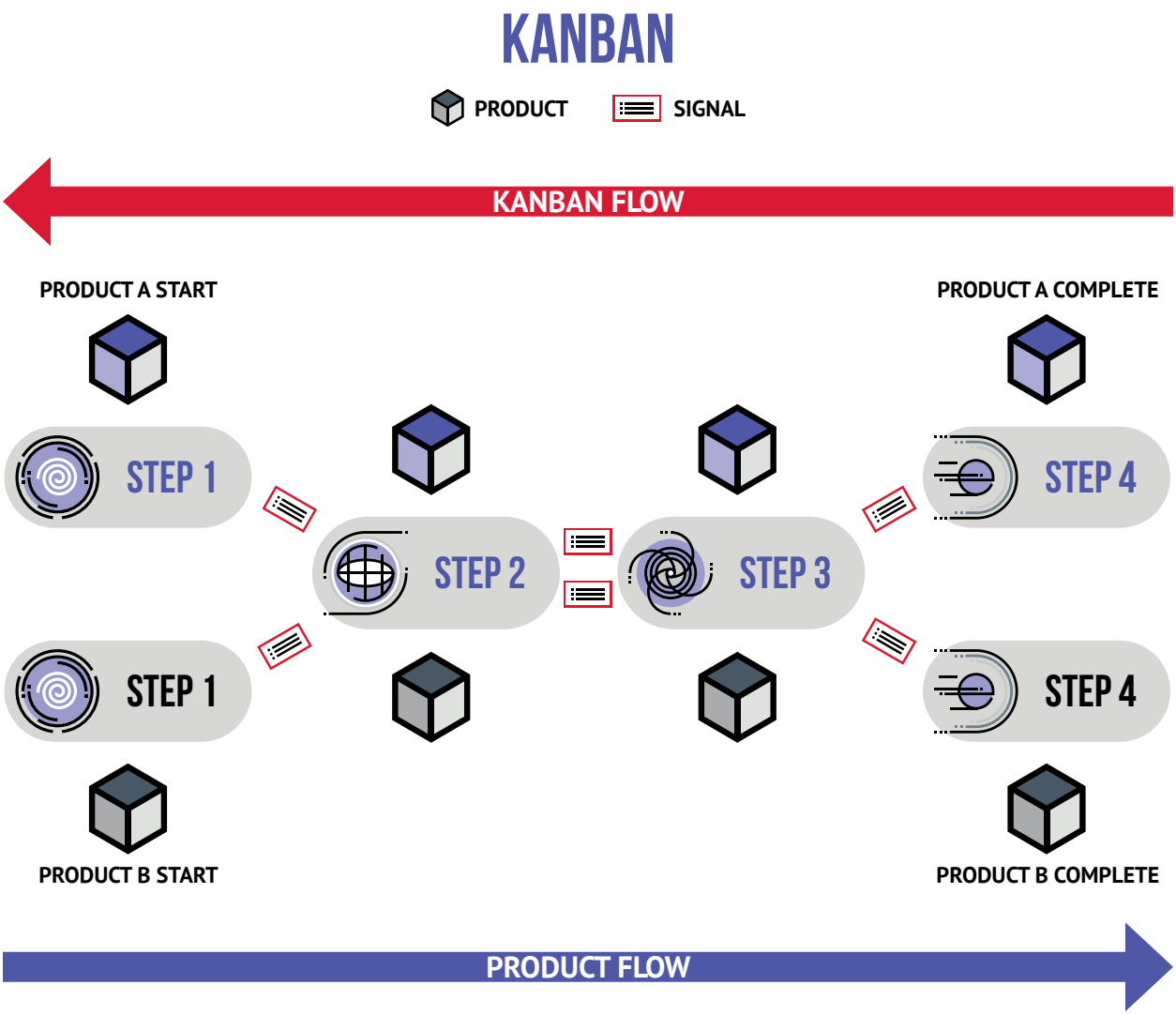


For manufacturing processes, this approach tends to highlight the waste associated with having too much inventory or work in process. For manufacturing this waste was addressed by controlling how work was release into the process to limit inventory. Henry Ford did this for high volume manufacturing of one product by creating a completely linked assembly line with limited room for excess inventory. Goldratt's used "Theory of Constraints" and "Drum buffer Rope." Toyota used Kanban.

Goldratt starts by finding the constraint or bottleneck in the process by looking at the cycle-time and batch size of each step to find the effective cycle-time and bottleneck for the process. He would create a small buffer in front of the bottleneck to ensure it was never starved for work. Then he linked the release of work at the front of the process with a rope tied to the bottleneck and used the effective cycle-time of the bottleneck as the drum or rate for releasing work into the process as shown below limit excess inventory.



Taiichi Ohno further refined this process to ensure there were not short-term bottlenecks building up between different process steps with Kanban. He would limit the inventory between each process step with a tub or a card that would signal to the upstream process that until it got an empty tub or card back from the downstream process it shouldn't create any more products as shown below. Using this Kanban approach for managing flow he could ensure excess inventory and wait times were not building up in the system while ensuring the bottleneck was not starved for work.

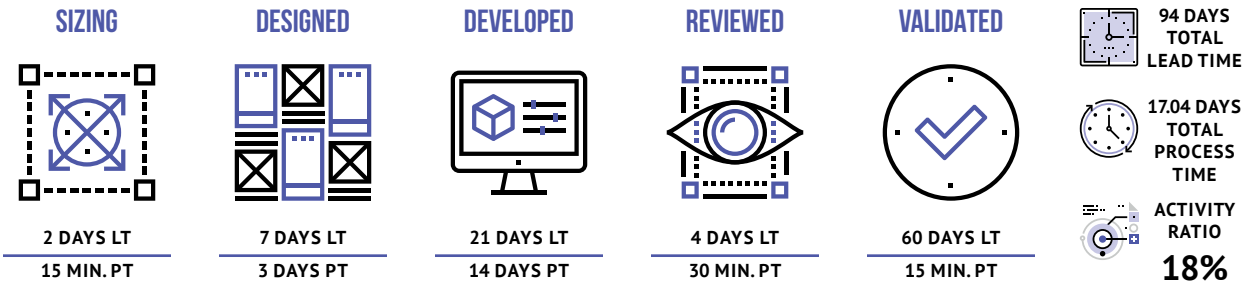


Value Stream Mapping has also been used extensively for business process re-engineering. A typical example would be re-engineering an insurance claim process. In this case you would map out all the steps and people required to process the insurance claim documenting process and lead time as shown below. This type of analysis tends to show waste associated with handoffs between different people as the biggest sources of waste as opposed to excess inventory. The improvement then typically required eliminating handoffs and moving to having one person manage the end to end process dramatically reducing wait time and get to a much higher activity ratio.

BUSINESS PROCESS VALUE STREAM MAPPING

The approach I frequently see now when organizations Value Stream Map their software processes is that they map a new request from idea to delivery. This is fairly straight forward to track a requirement in tools through all the different states and document how long it takes to move from one state to then next until it is released to the customer as documented below.

SOFTWARE VALUE STREAM MAPPING



This type of analysis frequently shows there is a fair amount of wait time and opportunities for improvement. The challenge I have with it is that it doesn't do a very good job pointing out everything that needs to be fixed like it does for manufacturing or business processes re-engineering. It could be too much work in process like in manufacturing that is causing the wait time between sizing, designed, and developed and we need to address how we release work into the system to avoid building up too much inventory. It could be handoffs between workers between developed and reviewed that needs to be addressed especially if the work environment is ticket driven. The classic approach to value stream mapping is very good at pointing out issues with these things and opportunities for improvement where the requirements are processed one at a time.

The difficulty I see with this approach is that it doesn't do a very good job of highlighting the waste associated in the value stream from when the requirement is complete and reviewed to when it is released to the customer for validation. This release process is frequently the biggest source of issues for most large companies and the classical approach of tracking requirements from request to delivery or validation doesn't do a very good job of highlighting the waste an inefficiencies in that process.

For the release process we need to move from tracking individual requirements to mapping how a group of requirements come together into a release because that is how the process works for most organizations. This release process takes all the requirements that are ready, batches them up, and runs them through the deployment pipeline for a release which is very different than business processes or manufacturing that handles one item or batch of items at a time. We don't have to worry about inventory building up in front of this process because the batch size can grow as large as required even though large batches might not be very efficient. Also the inefficiencies in the release process aren't typically slowed down by handoffs between different groups.

The types of things that slow down a release and create waste and inefficiencies are different. It might be an organization's Software Development Lifecyle that states until a certain milestone like "Development Complete" is reached requirements are not moved to the next stage in the deployment pipeline. It might be that organizations have to spend extended time and energy in hardening phases to inspect in quality because they have not shifted to building in quality. It might be waste associated with manually creating environments, deploying code, or testing that need to be automated to improve the efficiency of the release process. The problem I see when reviewing software Value Stream Mapping from organizations using the classical approach is that these types of issues are not identified.

We can use classic Value Stream Mapping for the steps in the process that have a fixed batch size as shown below but for the deployment pipeline used for the release process we need a different approach that is designed to identify the challenges that are unique to software. An approach that can leverage what has been proven effective in manufacturing and business process engineering over decades but can highlight the unique issues associated with releasing software.

SOFTWARE DEVELOPMENT PROCESS

IDEATION	DEVELOPMENT	DEPLOYMENT PIPELINE	TRIAGE
PROCESS CYCLE TIME DEPENDS ON INVESTMENT	**PROCESS CYCLE TIME** VARIABLE FOR EACH REQ.	**PROCESS CYCLE TIME** FIXED ONCE AUTOMATED	**PROCESS CYCLE TIME** DEPENDS ON BATCH SIZE
BATCH SIZE 1	**BATCH SIZE** 1	**BATCH SIZE** → ∞	**BATCH SIZE** → 1
EFFECTIVE CYCLE TIME DEPENDS ON INVESTMENT	**EFFECTIVE CYCLE TIME** VARIABLE FOR EACH REQ.	**EFFECTIVE CYCLE TIME** → 0	**EFFECTIVE CYCLE TIME** AS SHORT AS POSSIBLE

That was the goal of "Engineering the Digital Transformation." I took everything I had learned identifying waste in large organizations to provide a framework that goes beyond classical Value Stream Mapping to provide a systematic approach for analyzing a broad range of applications based on software. After using this approach for several years my experience is that it highlights issues that just don't show up in the software Value Stream Maps I am seeing in the industry. Call them common mistakes or just the need to think differently about how to identify opportunities for improvement in the software development and delivery process.

For over a decade, one of the most effective tools I have used to engage with clients has been the *Value Stream Mapping* assessment. I have done 100s of Value Stream Mapping assessments over this time, working with organizations across the maturity and size spectrum. From startups struggling to survive, to traditional mid-size companies fighting off upstarts, to global enterprises looking to become innovative, technology driven organizations. Value Stream Mapping, if done right and with the right stakeholders, always exposed the maturity or lack thereof of the organization. In this paper Gary Gruver, author of *Engineering the Digital Transformation* shares the common mistakes and challenges he has seen when applying Value Stream Mapping techniques to software delivery. Gary is one of the few people I know who truly gets enterprise scale transformation of software delivery. He has led them on both sides of the table — as a leader within the enterprise, and as a consultant and coach to the transformation leader in the enterprise. His experiences with Value Stream Mapping are right here for all readers to learn from and apply.

—Sanjeev Sharma
Principal Analyst, accelerated strategies

Gary Gruver is an Accelerated Strategies Group Analyst, Gruver Consulting CEO, an acclaimed author and an in-demand speaker. Gary brings a proven track record of transforming software development and delivery processes in large organizations.

For more information, click here.