**Summary Analysis of “The Learning Organisation” w/ref to S//W development.**



**Definitions of related terms:**

**Agile** software development refers to a group of software development methodologies based on **iterative development**, where **requirements** and **solutions** evolve through **collaboration** between **self-organizing cross-functional teams**.

**Kaizen** is the practice of c**ontinuous improvement.** Today Kaizen is recognized worldwide as an important pillar of an organization’s long-term competitive strategy. Kaizen is continuous improvement that is based on certain guiding **principles**:

* Good processes bring good results
* Go see for yourself to grasp the current situation
* Speak with data, manage by facts
* Take action to contain and correct root causes of problems
* Work as a team
* Kaizen is everybody’s business
* And much more!

One of the most notable features of kaizen is that big results come from many small changes accumulated over time. However this has been misunderstood to mean that kaizen equals small changes. In fact, kaizen means everyone involved in making improvements. While the majority of changes may be small, the greatest impact may be kaizens that are led by senior management as **transformational projects**, or by **cross-functional teams** as kaizen events. The core idea is to maximize customer value while minimizing waste. Simply, lean means creating more vaOne def for anthro said it covers past, present and future though... too many definitions online I think.lue for customers with fewer resources.

A **lean organization** understands customer value and focuses its key processes to continuously increase it. The ultimate goal is to provide perfect value to the customer through a perfect value creation process that has **zero waste**. To accomplish this, lean thinking changes the focus of management to looking at the system as a whole, eliminating waste along entire value streams, instead of at isolated points. This creates processes that need less human effort, less space, less capital, and less time to make products and services at far less costs and with better quality, compared with traditional business systems. Companies are able to **respond to changing customer desires** with high variety, high quality, low cost, and with very fast throughput times. Also, information management becomes much simpler and more accurate.

A **poka-yoke** is any mechanism in a **lean** manufacturing process that helps an equipment operator **avoid (yokeru) mistakes (poka)**. Its purpose is to **eliminate defects** by preventing, correcting, or drawing attention to human errors as they occur.

People often find programming concepts and structures difficult to grasp. The only way to get better is to practise. Sometimes it can be difficult, but the programming areas of the brain, like a muscle, can improve with exercise. Cycling a bike, typing a shoelace, doing the Macarena- these can all be difficult at first, and less difficult after lots of practise. We can learn to adapt, improve, and change.

Humans are designed for learning. Natural curiousity => learning can be enjoyable (intrinsic motivation). Our prevailing system of management has destroyed our people,” The way we “teach” can destroy motivation, and “avoiding mistakes” can mean an unwillingness to make a mistake, resulting in “mediocre performance”. The impulse to learn, at its heart, is an impulse to be generative, to expand our capability. This is why leading corporations are focusing on generative learning, which is about creating, as well as adaptive learning, which is about coping. Generative learning would mean that we can see a demand for something that doesn’t exist yet: Steve Jobs is a prime example.

*“Show me someone that has never made a mistake, and I’ll show you someone that has never achieved anything”.*

Companies are born and die in the same way as people, but some have very short lives. Some can live over a century. “Interestingly, the key to their survival was the ability to run “experiments in the margin,” to continually explore new business and organizational opportunities that create potential new sources of growth.”

What kind of company can facilitate “integrating thinking and acting at all levels”? Learning organizations emphasizes increased adaptability. Traditional authoritarian bureaucracies often responded too slowly to survive in changing business environments.

The Japanese developed the “Toyota Production System” and were able to outperform the Americans by looking at the entire process differently. Part of this later developed into TQM. A core definition of total quality management (TQM) describes a management approach to long–term success through customer satisfaction. In a TQM effort, all members of an organization participate in improving processes, products, services, and the culture in which they work. This can require some creative effort (generative learning) to look at things from a different perspective.

*The Japanese saw the significance of delays because they saw the process of order entry, production scheduling, materials procurement, production, and distribution as an integrated system.*

So, while we can break systems down into smaller pieces (modularisation) and solve (or redesign, or “refactor”) each part independently, we also need to be able to look at the system as a whole. A problem in one part of the system could potentiality cause a symptom in another system. Good design can avoid this. When we have objects (sprites) in a game, we know that they have their own code, but they may often interact with other objects (a ball with a paddle) and we cannot look at one, without also looking at the other when looking at the entire system (a game of Pong).

The Japanese also developed closer relationships with all the stakeholders in the entire production process. In the same way, Agile teams will have greater and closer co-ordination to reduce time delays in the development of a system.

*Agile software development refers to a group of software development methodologies based on* ***iterative development****, where* ***requirements*** *and* ***solutions*** *evolve through* ***collaboration*** *between* ***self-organizing cross-functional teams****.*

Managers walk the well-worn path, ordering subordinates to follow (at gunpoint).

Leaders walk the path they make themselves, and others follow.

If you were to set a vision for where you would be in 5 or ten years time, what would that be? What are the steps you would need to take to make that a reality? What time frame would you put it in? What that the steps needed to achieve each milestone (major point on the road). If the analogy were a game, and completing the game required clearing five levels, you would need to have a target by when to clear level one. Each time you get killed, you might learn something new, and try to avoid the mistake the next time. Can you set a vision for yourself?

Vision + understanding of current reality => creative tension. Many people & orgs find themselves motivated to change only when their problems are bad enough to cause them to change. This works for a while, but the change process runs out of steam as soon as the problems driving the change become less pressing. With problem solving, the motivation for change is extrinsic. With creative tension, the motivation is intrinsic. This distinction mirrors the distinction between **adaptive** and **generative** learning.

*“Leadership is inter-twined with culture formation”* so if you want to have an organisation that is “agile”, you need the appropriate structure as well as suitable leaders. **Leader as designer** rather than “the captain” is an interesting one. The analogy is that you cannot have an agile organisation unless you have designed it to be able to be agile- to be able to respond and adapt quickly.

*“The first task of organization design concerns designing the governing ideas of purpose, vision, and core values by which people will live.... the second design task involves the policies, strategies, and structures that translate guiding ideas into business decisions.*

So, there is more to an organisation than moving boxes around and writing a rule-book. *Only when the planners re-conceptualized their basic task as fostering learning rather than devising plans did their insights begin to have an impact.* Scenario Analysis gets people thinking about what they would do under different possible outcomes. Similarly, when planning software, you have to consider what will happen under different scenarios, and “ferreting out the implications”. Trend Analysis: Many new products are becoming “smarter”. Could this trend influence software design?

Skills can be developed: some of the more important skills include:

* Developing a vision- like when rapidly prototyping a product, it means getting an understanding of what the final product will lok like- be it a game, app, website or whatever.
* Communication skills amongst team members
* It is an ongoing process, like the spiral model, it doesn’t stop after one pass

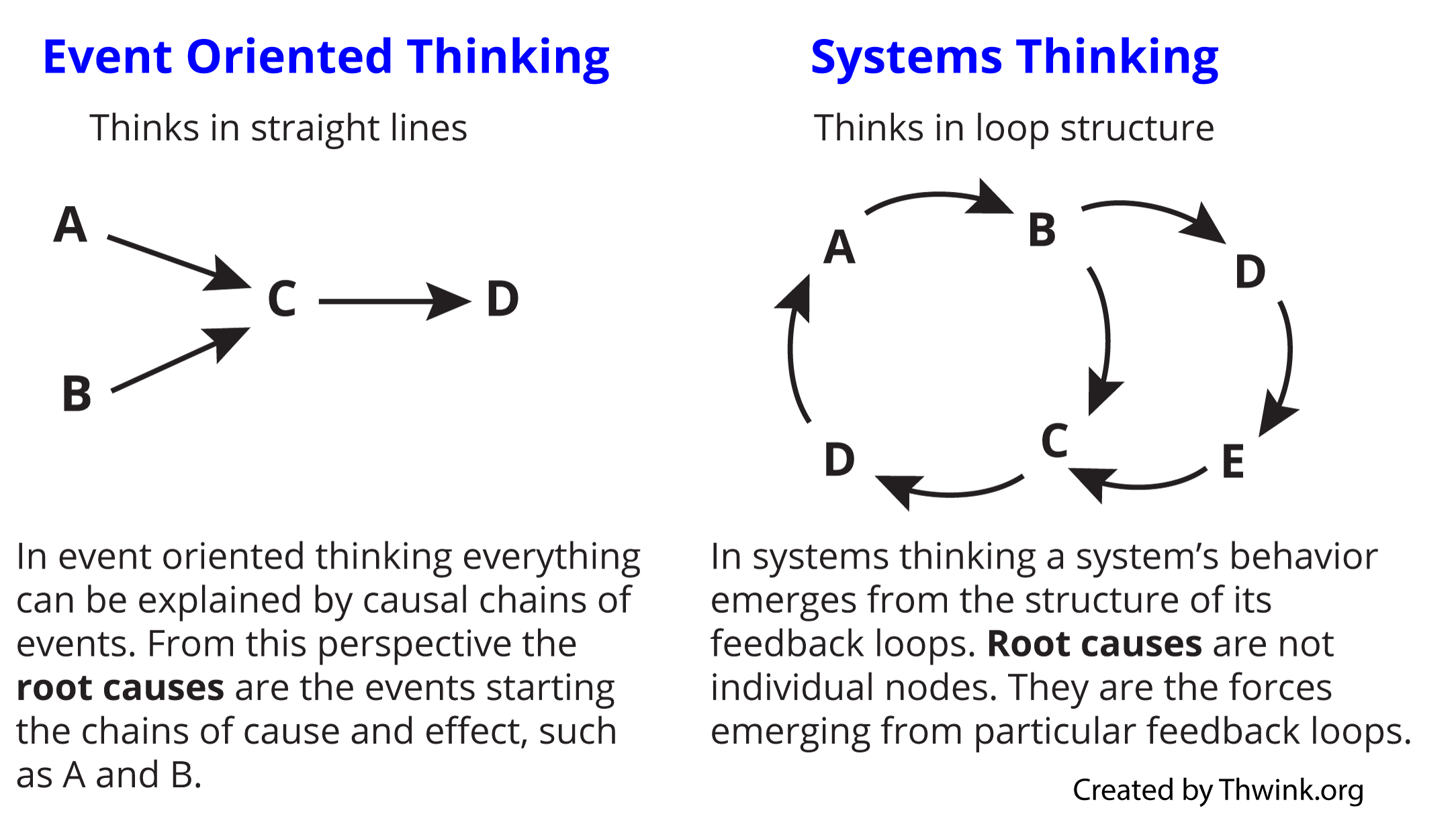
Do not make assumptions about what the customer wants- because they ask for item “x” does not mean they do not care about items “a-g”. Decisions should be based on objective reasoning. Facts and figures. Data in context. Objective data is not biased. Subjective information depends on the person: “Vindaloo Curry is not too hot”. It may be true to some, but not to others.

When working in teams, if someone disagrees with you, ask why- looking at objective data, and what they propose. Encourage more ideas to get better ones. Recognise that sometimes people do not say what they mean.

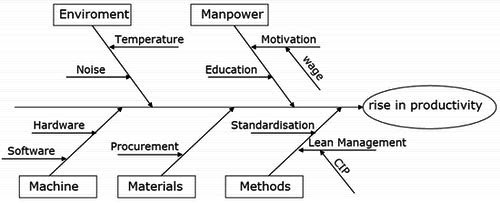
Systems thinking is a management discipline that concerns an understanding of a system by examining the linkages and interactions between the components that comprise the entirety of that defined system. It is seeing the big picture, and the relationships of those things inside.

In poka-yoke (Kaizen), one key idea when a problem is found is not to blame the person, but to look at the process that allowed it to happen- then change the process. You must also be careful when making changes, that they do not cause problems “down the line”. On the other hand, small changes can have huge benefits.

*Systems thinkers refer to this idea as the principle of “leverage.” Tackling a difficult problem is often a matter of seeing where the high leverage lies, where a change- with a minimum of effort- would lead to lasting, significant improvement.*



One method to identify problems in a manufacturing environment is the Ichikawa Diagram. One version of this (there are many) is shown below:

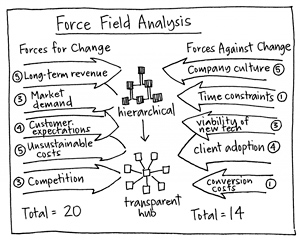


If we were to look at this in the context of a software development company, the top two would not change, but for machine we could consider: Is it a computer running Windows, Mac or Linux? Is the hardware powerful enough? Id there enough RAM? What Editors are we using to write code? What kind of testing are we doing (testing frameworks)?

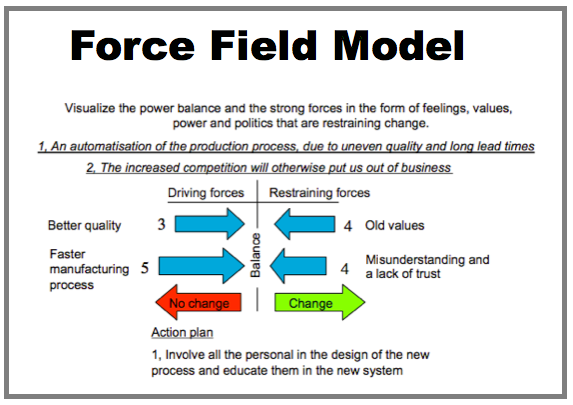
When we come to materials, well the product is built from code. Code can be created in many languages and for many platforms. Did we pick the right language? Has it been optimised? Could it run faster? Could it use less memory?

Methods may include how we manage to co-ordinate the members of the team, are we using waterfall or agile, are managing meetings properly, and so on. There has to be a balance between ensuring quality (checks and reporting) and the time it takes. A quick-fix can cause more problems later. Lowering standards is ultimately self-defeating. It is possible to have low cost and high standards, but it means looking at how to do this- it will take some time, it may take new technology, but it is possible. An example of this is “testing frameworks” which can be used to run lots of tests on a software product- automatically, so speed is increased and human error reduced. Better quality, lower cost!

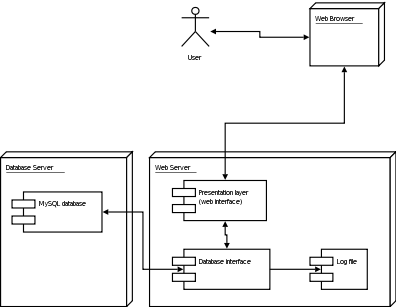
Looking at things in a different way leads to insights. One way, for example, to “View” the arguments for change in a company is Force-Field Analysis, and an example is shown below.



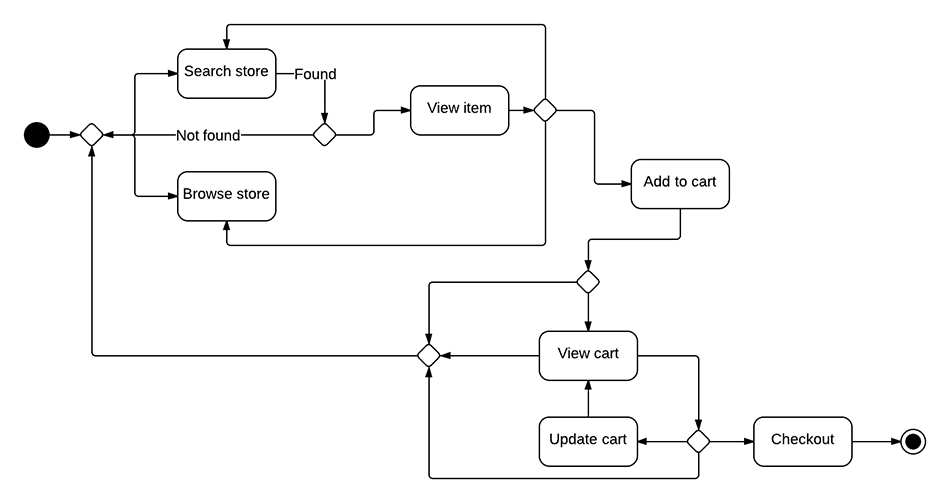
There are rules and guidelines easily available online for conducting a FFA.



Looking at a diagram of a SW development process is also easier to understand and see the connections. Looking at a system can also be modelled as below:



UML Deployment Diagram



UML Activity Diagram

