How the lean production methods described in the book can be applied to software development?

Following lists seven principles of lean production method which we can apply in software development.

1) Remove waste or things which are not adding value.

Recognizing waste in software development is a challenging task. We can adapt some the basic principles and guidelines in order to achieve this.

- Tasks Allocation: Divide tasks such that each resource is productive and not wasting too much time on dependent components or features. Just like in a multithreaded program if threads are waiting for each other, then we are not benefitting much from concurrency. Similarly if resources working in parallel are busy then productivity will go up.
- Proactive in communication: communicate about changes and failures so that others can adapt to it, instead of wasting time on working on things which sooner or later are going to be changed or aborted.
- Remove features/functionalities: Remove features or functionalities which are not used or which can be done in a more efficient way. We can remove code that are not being used in production or code that partially implements certain features.
- Requirement analysis: Analyze requirements such that we exactly know what to implement. During requirement analysis we can get rid of things which are not adding values and if possible add useful pieces. Bad requirement analysis can has a cascading effect. It is going to result into bad design and hence wrong coding and implementation. So requirement analysis is a very important step in software development cycle.
- Removing redundancy: Code reusability can lead to thin and high quality software. It brings consistency in the implementation. Redundant code is hard to maintain, error prone and hence can result into more production time and cost.
- Comprehensive testing: carefully designed testing can catch oversights. The oversights could have happened at any step of the development process including requirement analysis, designing, implementation or coding.

2) Build quality at each step

Defects are bound to happen in software development. The idea is remove defects at each step and more frequently rather than continue doing work and allowing defects to pile up and decide to fix defects at a later stage in the development process. Starting from requirement gathering we should collect feedback from users in order to come up with a solid requirement. Even at implementation stage code a small portion of the system and test it to make sure that even the smaller pieces are defect free. Leaving defects for longer periods can cause accumulation of more defects in the process and results into low quality and high development cost software.

3) **Delay in decision making**

In order to tackle uncertainties we can delay the process of decision making as late as possible. It's always a good idea to have something which can adapt changes. We can delay in finalizing requirements and we can do it when necessary so that we have more understanding and clarity over the subject or business. Agile or iterative methods will allow us to re-visit the pieces which have changed and a flexible environment will allow us to commit the changes in the software.

4) **Fast Delivery**

It helps in getting competitive advantage and customers need time to decide about their future requirements if they get what they require in the present. That gives development cycle the advantage of delaying their decisions.

5) **Build team**

Managers should bring positive environment which encourages innovation. People should have respect for each other. Giving feedback, supporting colleagues and subordinates and removing hindrances for team members are the qualities of a productive team. Managers should not control or micro manage resources working under him rather he or she should listen to the subordinates, listen to their ideas and feedback.

6) **System as a whole**

Large complex software can be broken into smaller modules and each module can be individually developed defect free or with minimal defects and can be integrated into a big system. Defining the associations and interaction between individual components will help large systems work smoothly and correctly. It is important to consider how an individual component or software will behave in a large integrated system.

7) **Integrity**

For customers the software should look like one system and not collection small components. Software should know what customers want and it should run correctly and smoothly.

What benefits can be derived?

The lean manufacturing methods are very relevant in software development world. The following benefits we can derive from lean method.

- Understanding customer and requirement will greatly help in defining and designing the product
- Defining the integrated environment will result in software which will work correctly and smoothly.
- Early time to market will give competitive advantage.
- Creating value in software will attract customers and make software more usable.
- Testing early and frequently will make software with minimal defects.

What are the commonalities between auto manufacturing and software development?

- Both intend to produce high quality products.
- Both intend to reduce production time and cost.
- Both require products to be tested.
- Both require design and implementation adaptive to changes.
- Both require customer involvement and feedback.

How can software quality be improved through the techniques described in the book?

Applying lean techniques can greatly improve quality of the software. Reducing wastes by following lean practices will create software with minimal defects. Understanding requirement correctly by customer's feedback and removing defects at each level will result in high quality software. Leaving defects in design and code for long is a very common cause of buggy software.

Apart from implementing individual components correctly, defining the interaction between components and their behavior in an integrated system is a very important factor in improving quality. Practices like getting rid of unused features and code can be of immense help. Testing frequently and individual components will contribute to large working software.

Effective communication and teamwork is also a very important factor.

What aspects of our software design course are relevant?

- Learned different software development process (Agile, XP etc.).
- Learned how to design individual components and their interactions and how they integrate into a larger system.
- Peer review acted as feedback and make necessary changes based on the feedback.
- Testing individual component and system as whole minimizes defects and helped in producing high quality mobile application which worked correctly and smoothly.
- Design reviews helped in having experience with working in a collaborative environment.