PMPC
2019
Project Management
Practitioners' Conference
PMPC
2019
July 12-13, 2019
NIMHANS Convention Centre - Bengaluru

Knowledge Driven Development (KDD) - Digital Knowledge Management for Digital Transformation

Capability enhancement

PMIBC-19-1-004

Manoj Kumar Lal

Author of the book 'Knowledge Driven Development'

CONTENTS

ABSTRACT

In a world which is going through major disruption driven by digital innovations and changing consumer's needs, the digital transformation of an organization's process and products is the need of the hour. If the necessary knowledge for this transformation is not available to the organization in real time basis the speed of the transformation may slow down. Many times, the knowledge may be spread across heterogeneous sources and prone to duplications, inconsistencies, and obsolescence.

I have come up with a new knowledge management framework that can digitise knowledge. Knowledge can be represented by an inventory and relationship format, like the atomic structure where electrons and protons are the inventory, and attraction and repulsion is the relationship. The framework covers three major forms of knowledge - domain, enterprise and project knowledge and scopes it in four, eleven and seventeen building blocks respectively. Knowledge across these forms having the same representation format encourages reusability that can bring in a higher level of maturity in knowledge management.

Digital transformation can speed up significantly when the project team has access to digital knowledge and digital technology at the same time. Implementation and management related activities of the project are greatly assisted by digital knowledge.

The paper summarises this framework in plain English interspersed with a case study and examples. The objective of this paper is to create an awareness and interest in this new framework so that it matures further to benefit the industry and the academia.

INTRODUCTION

The core concept of digital transformation has evolved over the years. Let us try to understand it by analysing the banking domain. Computerisation at different branches of the bank was an example of digital transformation where bank clerks did not have to spend lots of effort to match debits and credits at the end of each working day. This work was taken over by the banking software. The current wave of digital transformation emphasises on self-servicing by the account holders for most of the transactions without involving bank employees. Speed and knowledge about the change are crucial for the current age of digital transformation.

The importance of knowledge management cannot be overemphasised while creating a wide range of outputs varying from software products to building constructions. Creating the output can be via a set of implementation activities if the knowledge about the output is known as shown in the figure 1.

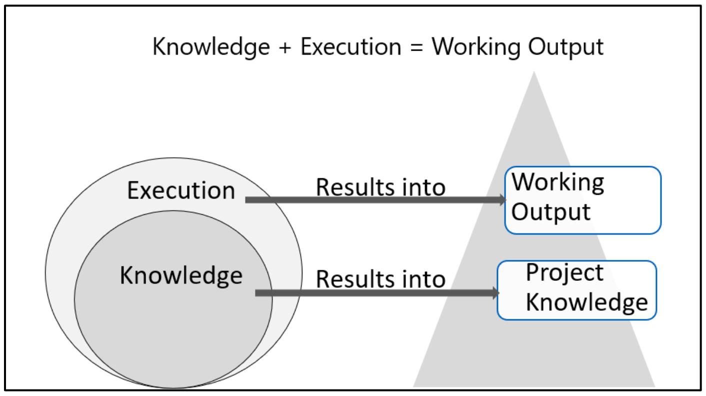

Figure 1. Working output: project Knowledge and its implementation

Figure 2 depicts the advancement of knowledge in four stages. The stage starts with analogue representation and evolves into a digital representation. Stage one deals with disseminating knowledge by getting others to memorise it and distribute it further. Stage two digitises it in the form of books and documents. Stage three converts them to soft copies and stores them in searchable portals. Most organisations are at stage three in their advancement of knowledge representation. This representation suffers from duplication, inconsistency and loss of relevance and hinders speed of change.

The fourth stage of knowledge representation deals with a specific format the knowledge should be captured in. This can be termed as knowledge 4.0 may be well suited to meet the needs of the digital transformation. This paper deals with Knowledge Driven Development (KDD), one of the knowledge representation formats claiming to be compliant to knowledge 4.0.

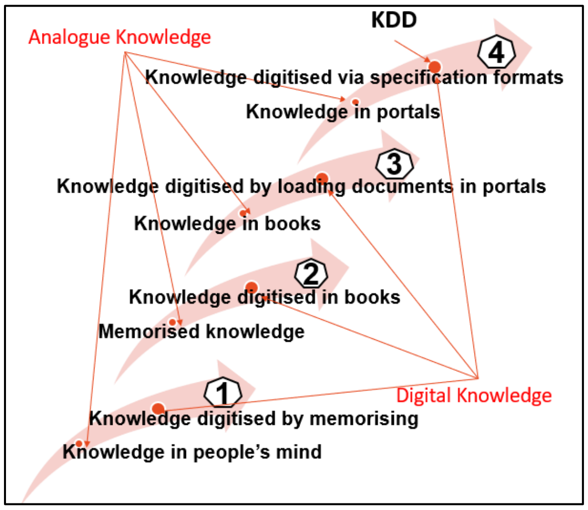
KDD
Analogue Knowledge
4
Knowledge digitised via specification formats
Knowledge in portals
3
Knowledge digitised by loading documents in portals
Knowledge in books
2
Knowledge digitised in books
Memorised knowledge
1
Digital Knowledge
Knowledge digitised by memorising
Knowledge in people's mind

Figure 2. Knowledge 4.0

DETAILS OF THE PAPER

Knowledge representation and its management has been a topic of discussion for a long time and has evolved particularly rapidly since the advent of computers. Following are some of the major forms of knowledge representation.

- User Story
- Use Case
- Entity Relationship diagram
- Data Flow Diagram
- Process manual
- Specification documents

Specification documents such as Business Requirement Specification (BRS) and Functional Specification Document (FSD) have wider scope and may contain other forms of knowledge such as User story, Business rule and Data Flow Diagrams. The specification documents are difficult to be kept updated in real time mainly due to the change management and sign off mechanism. This restricted the use of Waterfall methodologies that largely depend on specification documents for knowledge management. Agile methodologies rely more on User Stories but compromises on the exhaustiveness of knowledge captured (thereby knowledge remaining in the minds of the project team) and any attrition in the project team may become a challenge for smooth project delivery.

The above knowledge, when converted into a digital representation, helps overcome the challenges of the specification documents and User Stories. Knowledge Driven Development (KDD) provides the representation format to digitise the knowledge and rest of the paper details it.

KDD is first defined and explained via examples. A case study in insurance domain is described to visualise the concepts. At the end, different usage scenario of KDD is listed so that the advantages of using KDD can be better understood.

Knowledge Driven Development (KDD) – an overview

In general, from a graduating student to a working professional three types of knowledge are relevant.

- **Domain knowledge**: Knowledge about a specific domain such as insurance, retail.
- **Enterprise knowledge**: Knowledge about an organisation operating in one or multiple domains. An example is knowledge about an insurance company working in insurance domain. It includes knowledge contained in its past and present product brochures and process manuals used by different departments of the company.
- **Project knowledge**: Knowledge about an initiative in the form of a project whose ultimate objective is to enrich the enterprise knowledge of an organisation to keep it competitive in the marketplace. Knowledge about a new product to be launched by the insurance company is an example of project knowledge.

KDD digitises domain, enterprise and project knowledge by scoping it in four, eleven and seventeen building blocks as seen in the figure 3 and listed below:

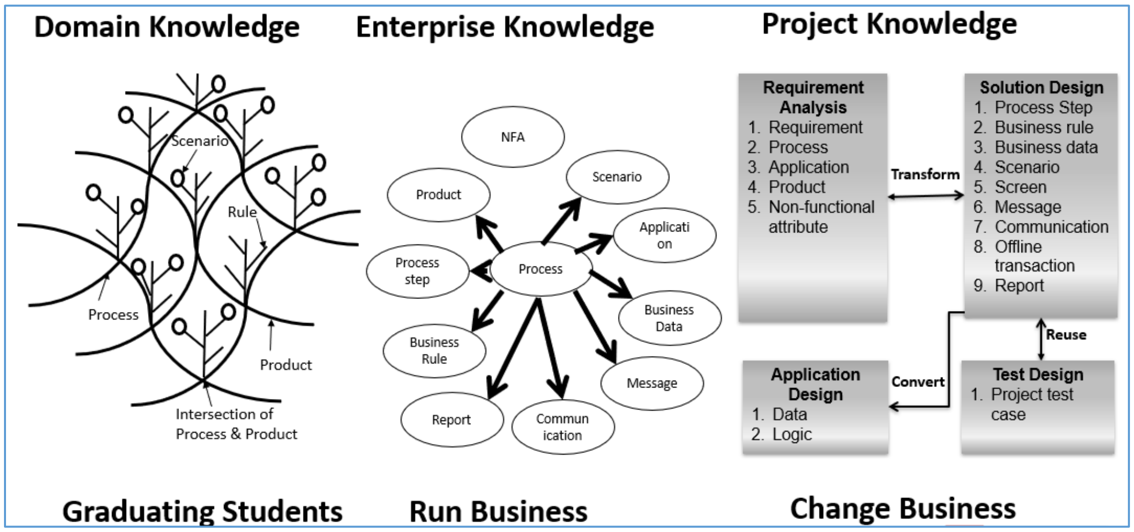

Figure 3. Scoping of domain, enterprise and project knowledge

1. Business rule
2. Scenario
3. Product
4. Process
5. Non-functional attribute
6. Process step
7. Business data
8. Report
9. Communication
10. Application
11. Message
12. Offline transaction
13. Screen
14. Requirement
15. Project test case
16. Data
17. Logic

A set of business rules and scenario at the intersection of products and processes consist of domain knowledge. Scenarios are used to help specify business rules. Rules for validating claim notification for an insurance product is an example of domain knowledge.

Enterprise knowledge for an organisation is created by customising the related domain knowledge and bringing sensible automation to it. Building blocks such as reports, messages, communications, non-functional attributes, business data and process steps are part of customisation and IT application is part of sensible automation. Rules, scenarios, products and processes of the domain knowledge are also subjected to customisation before moving from domain knowledge to enterprise knowledge.

For the project knowledge, relevant portion of the enterprise knowledge is extracted and customised. Six more building blocks are added for the completeness of the project knowledge taking the total number of building blocks to 17. They are:

a) Requirement and Project test case building blocks to scope the project and a mechanism to accept the project delivery.

b) Screen and Offline transaction building blocks that are heavily impacted for a project related to online and offline oriented works respectively.

c) Data and Logic building blocks that constitute application design, an important constituent of the project delivery.

There is an interesting observation to make about the project knowledge where the 17 building blocks are rearranged into four compartments representing four perspectives of the project knowledge as shown in the figure 3 above. They are:

i) Requirement analysis – project knowledge from the viewpoint of business analyst

ii) Solution design – project knowledge from the viewpoint of business system analyst

iii) Application design – project knowledge from the viewpoint of system analyst

iv) Test design – project knowledge from the viewpoint of test analyst

These are the four forms of the same project knowledge and as such complete. For example, if one has knowledge of all the test cases of the project, they will not find any new piece of knowledge from the other compartments of knowledge such as requirement analysis, solution design and application design.

These four knowledge compartments are related to each other. Requirements are **transformed** or expanded into solution design detailing more about the software to be built. Solution design is **converted** from business language to technical language in application design. Solution design details are **reused** in test design.

As stated earlier in figure 1, other than knowledge, a set of execution activities based on the knowledge results in delivery of the project. The set of implementation activities in this context are coding based on the output of the application design (i.e. data, logic) and test execution and defect management based on the output of the test design (i.e. project test cases).

Having scoped domain, enterprise and project knowledge into building blocks, lets understand how knowledge in the building blocks can be represented. Knowledge of the building blocks is represented in the inventory relationship format as shown in the figure 4 below.

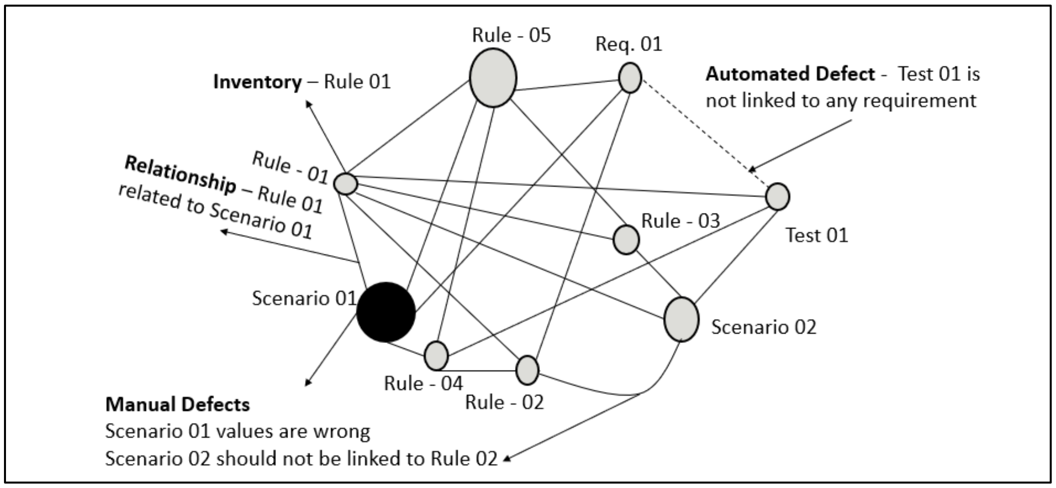
Rule - 05
Req. 01
Inventory – Rule 01
Automated Defect - Test 01 is not linked to any requirement
Rule - 01
Relationship – Rule 01 related to Scenario 01
Rule - 03
Test 01
Scenario 01
Scenario 02
Rule - 04
Rule - 02
Manual Defects
Scenario 01 values are wrong
Scenario 02 should not be linked to Rule 02

Figure 4. Inventory relationship format of knowledge representation

Knowledge is catalogued via inventories of building blocks. For example, 'Req. 01' is inventory of requirement building block being specified via attributes such as requirement id, requirement name and requirement description. Linkage between two inventories is known as relationship. An example can be: 'Req. 01' is related to 'Rule – 05' (inventory of Business rule building block). Relationship can also be qualified such as 'Req. 01' is partly satisfied with 'Rule – 05'. This representation format is the most stable representation format with examples varying from atomic structure to solar system.

Three important characteristics of this representation format are worth mentioning. The first one is the flexibility of the format to meet a specific need. If the project is time bound and has budget constraints, the number of building blocks required can be reduced along-with reduced attributes of the building blocks to produce an MVP (Minimum Viable Product).

The second characteristic of this representation is the way it assists in quality assurance. There are many types of relationships where the representation can assist in finding out a potential defect. For example, the relationship between requirement and project test case building blocks. Lack of relationship of a test case with requirements is a potential issue. Either the test case needs to be linked to a requirement or the test case is redundant and needs to be deleted. As shown in the figure 4 above, 'Test 01' is not linked to any requirement inventory and this is a potential issue. Another aspect of quality assurance is manual defect where inventories and relationships are manually reviewed, and any defect is tagged to inventory or relationship so that they can be addressed appropriately.

The third characteristic of the representation deals with extreme quantification of project delivery. Project Knowledge is represented in 17 building blocks and 153 (17*(17 + 1)/2: 136 inter building block relationships and 17 intra building block relationship) types of relationship between them. Relationship between requirement and product is an example of inter building block relationship and relationship between inventories of requirement is an example of intra building block relationship.

Every inter relationship gives rise to two potential source of failure and intra relationship one potential source of failure. For requirement and project test case relationship, the two potential sources of failure are: A project test case not linked to any requirement and a requirement not linked to any project test case. For requirement intra relationship the example is – A requirement not linked to any other requirement. This gives rise to 289 (2*136 + 17) potential defect types derived by the knowledge representation. There are also 170 (17 building block + 153 relationships) manual defect types. An example is defect in one of the requirement inventories.

To summarise, the extreme quantification of the project delivery is achieved in KDD by:

- 17 building blocks
- 153 relationships
- 289 automated defect points
- 170 manual defect points

Implementation activities are driven by data, logic and project test case building blocks and may easily be quantified for implementation and defect management (quality control). Project, knowledge and its quality assurance, implementation of the project knowledge and its quality control can be measured via related sets of draft, review and rework activities bringing end to end quantification to the project delivery.

This gives rise to a new project delivery methodology named as Knowledge Driven Development (KDD) based on digitised project knowledge. KDD with its digitised knowledge may also assist Waterfall and Agile methodologies.

KDD is not only a new project delivery methodology but a new way of managing project, enterprise and domain knowledge in a single digital format. End state visualisation of KDD is depicted in the figure 5 below (via a sample example) where these three forms of knowledge are digitised and are subject to seamless integration.

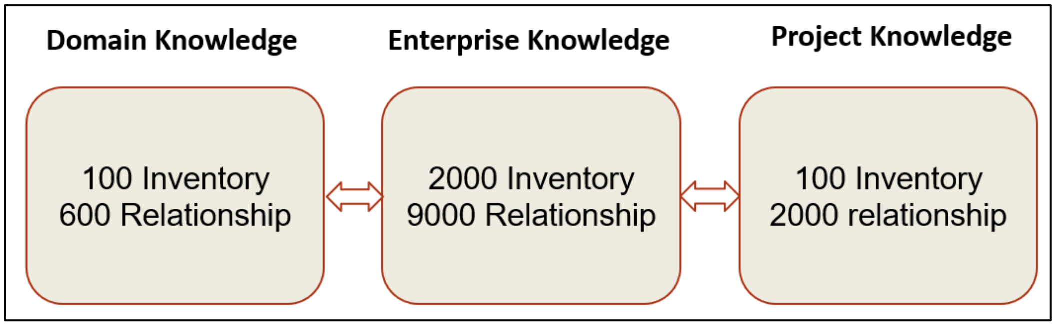

Figure 5. A sample of digital domain, enterprise and project knowledge

Let us understand domain, enterprise and project knowledge with an example of the inventory of business rule building block for password management process of portal domain.

- Business rule (domain knowledge): The password should be strong enough so that it cannot be compromised easily.
- Business rule (Enterprise knowledge): A company has decided that all its Passwords must have 8 characters consisting of a combination of alphabets, numeric and special characters.
- Business Rule (Project knowledge): There is a need to send an email to the user immediately after he/she changes their password.

KDD case study in insurance domain

Through a small case study in insurance domain, an attempt is made to visualise the end to end offering of KDD via domain, enterprise and project knowledge and its seamless interaction.

In the figure 6, a snapshot of investment related insurance products against policy administration processes is listed.

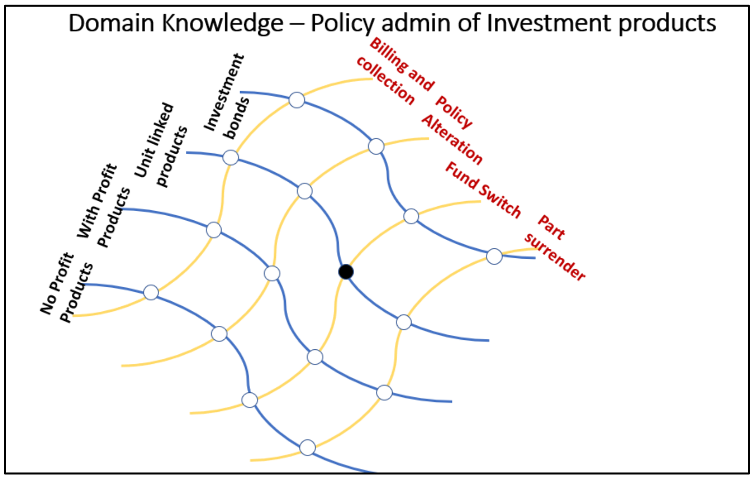

Figure 6. Policy administration processes of investment related insurance products

Fund switch process for unit linked insurance product is the product process intersection where domain knowledge can be captured as a set of business rules and scenarios. This is taken as scope for this case study. In the figure 7 below, fund switch is having process id of 11 and unit linked insurance product has the product id of 10. They are linked to 21 business rules and there are 10 scenarios assisting to specify these business rules completely.

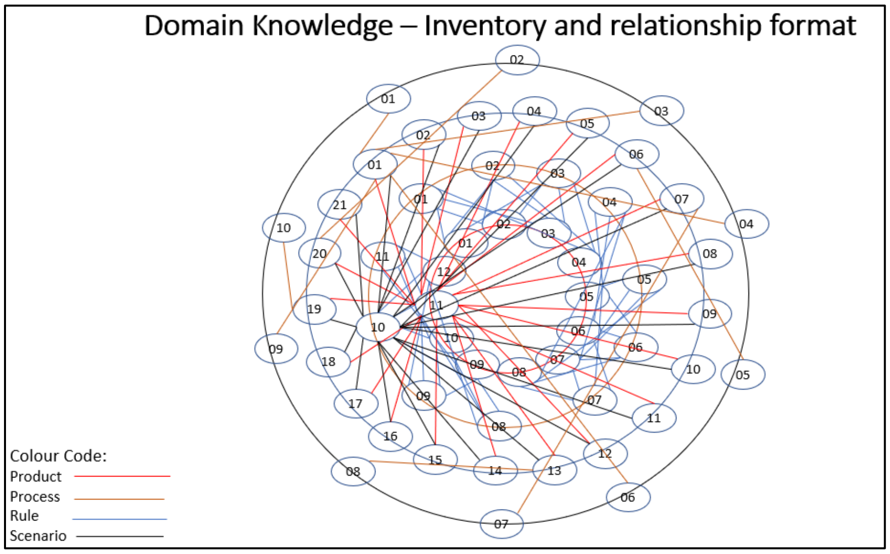

Figure 7. Domain knowledge of fund switch for a unit linked product via inventory and relationship

An example of business rule is:

Rule 01 - Target funds must be valid, must have at least one fund specified and total fund distribution of switched out amount must equal 100%.

And that of a scenario is:

Scenario 07 – Switch request channels: 1) Web, 2) Post, 3) Phone, 4) Email

This portion of the insurance domain knowledge – fund switch for a unit linked insurance product (and related domain knowledge needed for the context setting) can be represented in 54 inventories (12 products, 11 processes, 21 business rules and 10 scenarios) and 96 relationships.

In this case study, four operations are performed. Figure 8 drives the first three operations as detailed below.

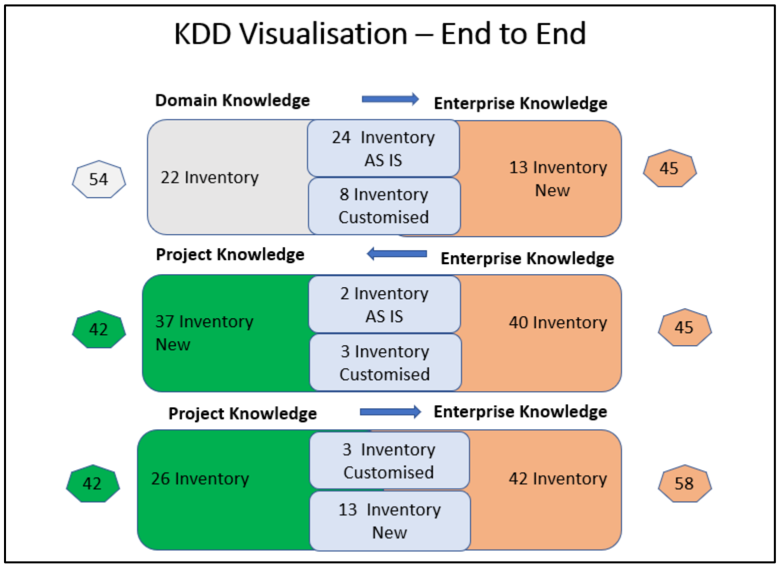

Figure 8. Fund switch – End to end visualisation of KDD concepts

1. Transition from domain knowledge to enterprise knowledge

Let us move to enterprise knowledge and assume a new insurance company is trying to adopt fund switch functionality for one of their unit linked products based on this available domain knowledge. It can be done in two steps. First the 4 building blocks of domain knowledge is customised to enterprise knowledge. An interesting observation here is – the overriding of switch charge is currently not available for the organisation and hence this rule cannot move from domain knowledge to enterprise knowledge. As a second step, seven new building blocks of enterprise knowledge are analysed for adding inventory and relationship of the core enterprise knowledge customised from the domain knowledge. The first diagram of figure 8 depicts the customisation of the domain knowledge to enterprise knowledge. For simplicity, only inventories are mentioned.

As shown in the diagram, 32 out of 54 inventories of domain knowledge participated in the customisation to the enterprise knowledge. 24 inventories were used on AS IS basis, 8 inventories were used post customisation and 13 new inventories were added to the enterprise knowledge mostly consisting of seven new building blocks that are not part of domain knowledge building blocks. An example from each of these three categories is listed below.

1. AS IS inventory: Rule - Fund switch must be initiated using a valid channel.

2. Customised inventory: Product - ABC Unit Linked Insurance Product (name of the product is customised to the organisation).

3. New inventory: Application - Legacy application where switch transaction is processed.

2. Transition from enterprise knowledge to project knowledge

As indicated earlier, the organisation lacks ability to override fund switch charges. Fund switch charges are applicable when the number of fund switches requested in a year exceed a certain number. Fund switch charges may be waived off, in case of a high net worth customer or as a part of limited period offer for all the customers. A project is initiated to allow fund switch charge override capability in the organisation. The second diagram of figure 8 shows that 5 out of 45 inventories of enterprise knowledge is reused for the project knowledge where 37 new inventories are added, mostly of six new building blocks of project knowledge that are not part of the enterprise knowledge building blocks.

An example from each of these three categories is listed below.

1. AS IS inventory: Application - Legacy application where switch transaction is processed.

2. Customised inventory: Rule - 10 free switches may be allowed in an year before it becomes chargeable. Overriding switch charges functionality should become enabled when it is 11th transaction in the current financial year.

3. New inventory: Rule - Fund switch charge override – reason must be selected from a combo box while creating the override request.

3. Enrich enterprise knowledge with project knowledge post project completion

Post project delivery, the ability to override fund switch is brought back to the enterprise knowledge. This happens by customising 3 inventories and adding 13 inventories to the enterprise knowledge as shown in the third diagram of figure 8.

An example from each of these two categories is listed below.

1. Customised inventory: Rule - 10 free switches may be allowed in an year before it becomes chargeable. Overriding switch charges functionality should become enabled when it is 11th transaction in the current financial year.

2. New inventory: Rule - If fund switch charge is overridden, the entire fund switch charge must be overridden, it cannot be reduced in part.

4. Demonstration of extreme quantification in project delivery

Let us drill down deeper into project delivery, being a core proposition of KDD. The figure 9 below provides a KDD view of the project knowledge, its quality assurance and implementation into a working software. For project management purposes, it is also split into measurable activities and effort spent is also tracked to provide various project delivery related metrices. The project knowledge in its digital format replaces the need for specification documents, user stories and acceptance criteria. The figure 9 is an implementation of the figure 1 where a working software was visualised by the project knowledge and its execution.

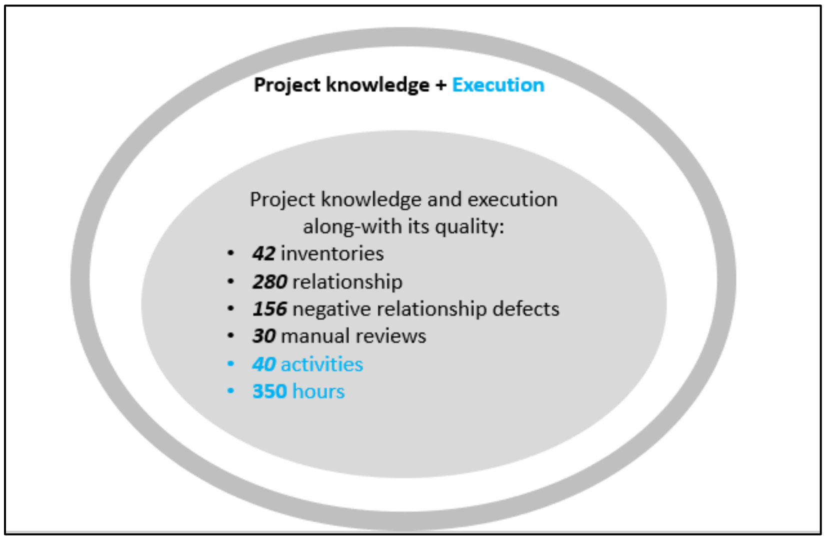
Project knowledge + Execution
Project knowledge and execution along-with its quality:
42 inventories
280 relationship
156 negative relationship defects
30 manual reviews
40 activities
350 hours

Figure 9. Software development – KDD approach of project knowledge and its execution

KDD – Usage scenario

KDD visualises a world where domain, enterprise and project knowledge are in the same format of inventory and relationship across domains. The digital knowledge creates a continuous learning environment where learnings are gained from the seamless interaction from the three forms of knowledge within a domain and across domains. Domains can be vertical where they are functional such as insurance and banking. Domains can also be horizonal with cross functional relevance such as portal and customer relationship management.

Overall KDD usage scenario are listed below:

- For a new organisation, the presence of digitised domain knowledge would be of significant assistance in setting up its processes and launching relevant products to the market.
- It is easier to learn about a domain through KDD when compared to learning it via books and documents. In KDD, digital domain knowledge consists of only four building blocks.
- Learning cross domain becomes easy as all the domains have the same four building blocks. At a generic level, they seem similar - entry, exit and maintenance seem high-level processes for most of the domains.

KDD usage scenario for project delivery are listed below:

- Every project contributes to the enterprise knowledge increasing reusability for the subsequent project. This creates an environment of continuous improvement in the organisation.
- Agile has a known gap in knowledge management and traceability that can be addressed by KDD. Also, the digital project knowledge can be kept maintained assisting the support team till the software is decommissioned enabling DevOps implementation.
- Project knowledge stored in the digitised format with exhaustive traceability greatly assists impact analysis.
- Due to digitisation the information is transparent compared to specification documents and the probability to detect the defect in the phase where it has originated is increased.
- Digital project knowledge and related implementation activities are quantified allowing different project progress related metrics to be obtained by the information contained in KDD as illustrated in figure 9.

- In KDD, once inventory and relationships are first drafted, any update to it can be linked to manual and automated defects allowing a record of change for all the future updates to the digital knowledge. This may help significantly in any root cause analysis and audit.
- Through four building blocks of requirement analysis phase, the project is scoped and requirements are linked to their inventories. These requirement inventories are transformed to one or more of the 9 building blocks of solution design ensuring natural evolution of requirement into solutioning.
- KDD has a structured format for solution design. These are easily traced to requirement inventories. Solution inventories arranged by application help in application design and solution inventories based on process help in test design.
- KDD assists in scientific evolution of application design. Application design inventories are created from solution design inventories. Completeness of application design can be verified by tracing them against requirement analysis and test design inventories.
- In KDD, test cases are created by reusing solution design inventory driven by process or scenario. It is then traced with requirements and application design inventories ensuring coverage of test cases.
- KDD consolidates various types of test cases in a single format during test design and ensures its optimal distribution for test execution. This ensures better quality test cases with reduced overall effort.
- In KDD, the application design inventories are stable as they are already traced to requirement, solution and test cases. Coding may be of clerical nature as the complexities are resolved at the application design phase.
- Test case failures in KDD should be significantly reduced when compared to Waterfall and Agile methodologies. Test cases are statically tested against application design inventories before execution.

CONCLUSION

After going through the details of the paper, it is easy to visualise that KDD is a nascent but strategic idea based on stable concepts that has the potential to become a widely used framework in near future. It has multi-dimensional applicability potential as summarised below.

1. An inter disciplinary course on domain knowledge based on KDD may be designed to reduce the gap between industry and academia. It may also assist in skill development initiatives. Such practical implementation of knowledge management will also help regaining confidence about knowledge management in industry and academia.

2. In the area of IT, KDD is a new project delivery methodology based on its digital knowledge management concepts.

3. KDD can be one of the futuristic enterprise knowledge management technique bringing in lots of efficiencies when compared to the traditional techniques.

4. The same format of domain, enterprise and the project knowledge may bring in increased and simplified reusability bringing in continuous improvement environment in an organisation.

KDD has the potential to accelerate digital transformation with its knowledge management proposition. To achieve this objective, there is a need to discuss about KDD at various industry and academic forums, its critical review so that it matures naturally. Also, there is a need to have further case studies, proof of concepts, prototypes, tools where one can experiment and visualise how KDD can assist them in their work from knowledge management perspective. KDD concept detailed in this paper is influenced by the book [1] of the same name 'Knowledge Driven Development'.

REFERENCES

Acknowledgement

I thankfully acknowledge my friends, relatives and colleagues who helped in increasing the quality and readability of this paper.

- Manish Choubey
- Sumi Lal
- Sushant Kumar Pati
- Ajay Rajhans
- Ambuj Kumar

References

1. Lal, M. (2018). *Knowledge Driven Development: Bridging Waterfall and Agile Methodologies* (Cambridge IISc Series). Cambridge: Cambridge University Press.