Luxoft
A DXC Technology Company

Five pillars of knowledge management for software engineering

by **Balaji Venkatramani**
Head of Digital Delivery Strategy and Solutions for
BCM APAC Luxoft India

Despite modern management solutions, it has remained a major challenge for decades. This is because of:

- Banks' reliance on large, legacy technology estates and application portfolios, plus inconsistent knowledge capture
- Staff churn and the consequent loss of knowledge
- Complex integration across a multitude of in-house systems, third-party products and hybrid solutions
- Unreliable capture of the incremental knowledge gained through testing, user acceptance, production, warranty and maintenance

Knowledge management – growing maturity

Level 1
(defined)

- **Defined guidelines exist for documents representing knowledge**, their coverage and frequency of updates required
- **All documents are version controlled** in a searchable repository
- **Traceability** to productionized changes exists
- All projects **follow the defined guidelines** consistently

Level 2
(repeatable)

- **Knowledge contextualized into themes** (business, process, technology), repeatable across projects
- Enhanced coverage through **automated documentation processes** consistent across projects
- **Reactive knowledge acquisition** per theme based on the specific needs of the project

Level 3
(measured)

- **Continuous knowledge management** through a metrics-based feedback loop
- **Proactive knowledge acquisition** leading to graded expertise levels enabled by theme-based academy
- **Proactive knowledge-management-enabled** autonomous squads

Higher maturity levels cumulate the practices of lower maturity levels

**So, what are the best practices for knowledge management?
Here are five points to consider:**

1: Knowledge capture needs to be prioritized and extensible

Gone are the days when software development began at inception for every product or application. Today, many products are extensions of previous products, or they're built in partnership with third-party products, or leverage off-the-shelf coding solutions and so on.
A prime requirement is to establish a robust knowledge transition process to make sure that teams have access to this historical knowledge before they attempt the future building of a product.

This knowledge transition process needs to ensure that:

Knowledge capture is prioritized. Not all historical knowledge needs to be captured at the time of transition and can be prioritized as follows:

- Visible book of work for a 6-month period
- Analysis of most frequent areas of software change over the last 6 months
- Analysis of most frequent areas of support in the last 6 months if already in production

Knowledge capture is extensible to augment the knowledge base in the future as required. This can be achieved by:

- Building a clear inventory of knowledge sources as metadata (documents, Wiki pages, development management tools like JIRA, source-code libraries, application logs, support ticketing systems, etc.)
- Mapping knowledge sources to relevant functional and technical modules of the product

2: Known knowns make up only half of the required knowledge

The other half is probably locked in the minds of previous team members or implemented systems on the ground.

Knowledge falls into one of three types:

Explicit
Visible and clearly available in known sources

Implicit
Knowledge gained from practical application of explicit know-how

Tacit
Knowledge that is understood, assumed or accumulated over time

Capturing implicit knowledge:

- **Practical assessment:**
  Study of implemented systems. For instance, code reverse-engineering tools can be used to develop documentation from implemented code

- **Shadow incumbent experts:**
  Observe or work alongside to gain insights

Capturing tacit knowledge:

- **Analyze** historical chat transcripts, emails, application logs, development and support ticket history and so on, based on relevance and reflecting real-time behaviors exhibited in history. You can use Natural Language Processing solutions to analyze unstructured knowledge sources and derive meaningful insights

3: Knowledge management needs quantification

Team knowledge levels are the most proactive indicators for expected quality of software and product. So, you need to develop an objective view of team knowledge, and a solid plan to aid its gradual improvement. “Knowledge index (KI)” is the primary metric.

Measuring the KI:

- **Proficiency-based KI:**
  The most objective approach is the Dreyfus model’s five levels of proficiency (novice to expert)
- **Objective KI:**
  Certifications, internal assessments, deliverable reviews, etc.
- **Team or squad KI:**
  Granular level measurement — specific to teams
- **Organizational baselines:**
  For common areas of knowledge e.g., specific domains or technologies
- **Target versus actual:**
  Baseline current KI, establish target KI and have an objective plan to bridge the gap between actual and target levels

Other objective metrics, such as software quality, efficiency and productivity, can provide a retrospective view of knowledge also. An ongoing corelation analysis between these metrics will produce the most accurate quantitative knowledge levels of the teams.

4: Knowledge is revised continuously

Knowledge management is a continuous journey. Fresh knowledge might not be captured in full due to time and cost pressures — you need to capture this knowledge before it becomes history.

Minimize the cycle by:

Incentivizing fresh knowledge capture:

- Convert collaboration tools like Confluence and Miro into knowledge repositories
- Deploy knowledge automation tools such as code reverse-engineering to generate documentation from code

Simplifying incremental knowledge capture:

A lot of knowledge is gained post-development of code and testing onwards. Use defect management and ticket management tools to record fresh knowledge whenever gained

5: Knowledge extends beyond people

Although knowledge means people and people mean knowledge, we still need to decouple the two.

Key person dependencies can cause stress which impacts work-life balance and affects team morale. You must download this knowledge to an offline repository and make it available for upskilling.

Offline knowledge repositories:

- **Recorded sessions:**
  Incentivize experts to make their knowledge available as offline recorded audio and video presentation**s**
- **Automated knowledge repositories:**
  Use code reverse-engineering tools and natural language processing solutions on implemented code
- **Collaboration tools:**
  Like Confluence and Miro

Organize these offline knowledge repositories into a structured, on-demand library with a "course curriculum". The team can assimilate this knowledge at their own pace, charting a career path for themselves based on their expanded knowledge.

If properly executed, knowledge management can become one of the greatest assets for banks, ensuring the successful delivery of products and services to their clients.

Get in touch

Luxoft help clients leverage our leading-edge knowledge-management practices to mitigate risk and grow their businesses.

Get in touch with **financialservices@luxoft.com** and find out how we can help you add business value through improved knowledge management.

Balaji Venkatramani
APAC Delivery Strategy
and Solutions Head

Balaji is a senior director with Luxoft India, heading the Digital Delivery Strategy and Solutions for BCM APAC. He has over 21 years' experience in the IT industry. Balaji has driven large-scale technology solutions and transformation initiatives in Silicon Valley technology companies as well as service partnerships with global financial clients.
He specializes in large-scale knowledge transitions, transformations, Agile, DevOps, big data and analytics, cloud and program management.