Chapter 6
Tools and Technologies for Architecture Knowledge Management

Peng Liang and Paris Avgeriou

Abstract As management of architectural knowledge becomes vital for improving an organization's architectural capabilities, support for (semi-) automating this management is required. There exist already several tools that specialize in architecture knowledge management, as well as generic technologies that can potentially be used for this purpose. Both tools and technologies cover a wide number of potential use cases for architecture knowledge management. In this chapter, we survey the existing tool support and related technologies for different architecture knowledge management strategies, and present them according to the use cases they offer.

6.1 Introduction

Architecting is a multifaceted technical process, involving complex knowledge-intensive tasks [195]. The knowledge that is both produced and consumed during the architecting activities is voluminous, broad, complex, and evolving and thus cannot be manually managed by the architect. Furthermore, such architectural knowledge (AK) [193] needs to be shared and reused among a number of different stakeholders, and across a number of the lifecycle phases. Especially as the size and complexity of systems increase, more stakeholders need to get efficiently involved in the architecting process and the knowledge management issues become quite challenging. The problem is exacerbated in the context of multi-site or global software development [75]. Finally the industry has also come to realize the need for efficient inter-organization AK sharing [76].

Therefore, the management of AK needs to be automated or semi-automated by appropriate tool support. This can be achieved similarly to traditional knowledge management tool support, by emphasizing the characteristics of software architecting. For example, tool support for architecture knowledge management (AKM) may

Peng Liang (✉) and Paris Avgeriou
University of Groningen, The Netherlands, e-mail: liangp@cs.rug.nl,paris@cs.rug.nl

M. Ali Babar et al. (eds.), *Software Architecture Knowledge Management*,
DOI: 10.1007/978-3-642-02374-3_6,

concern enforcing an architecting process, reusing architecting best practices, documenting architecture decisions, providing traceability between design artifacts, and recalling past decisions and their rationale. AKM tools can support a wide number of use cases, thus reducing the complexity of knowledge management in the architecting process and facilitating the knowledge-based collaboration of the involved stakeholders.

In knowledge management, a distinction is often made between two types of knowledge: implicit and explicit knowledge [234]; see also Chap. 4. Implicit (or tacit) knowledge is knowledge residing in people's heads, whereas explicit knowledge is knowledge which has been codified in some form (e.g. a document, or a model). Two forms of explicit knowledge can be discerned: documented and formal knowledge. Documented knowledge is explicit knowledge which is expressed in natural language or images in documents. Typical examples of documented AK are Word and Excel documents that contain architecture description and analysis models. Formal knowledge is explicit knowledge codified using a formal language or model of which the exact semantics are defined. Typical examples of formal AK models include AK ontologies [190] or AK domain models [10, 44, 325] that formally define concepts and relations, and aim at providing a common language for unambiguous interpretation by stakeholders. Organizations can employ three distinct strategies for managing their knowledge: *codification*, *personalization* [7, 143], and the *hybrid* strategy which combines the previous two [92]; see also Chap. 1. Figure 6.1 presents these different knowledge types in the vertical dimension combined with two knowledge management strategies in the horizontal dimension.

In this chapter, we first present a set of possible use cases that can be supported by AKM tooling. The set is not meant to be exhaustive; however it is well-grounded as it comprises a superset of the use cases either implemented or being in the wish-list of the existing AKM tools. In Sects. 6.2 and 6.3, we present existing tool support for the codification and hybrid knowledge management strategies, respectively. To the best of our knowledge there are no tools that support purely the

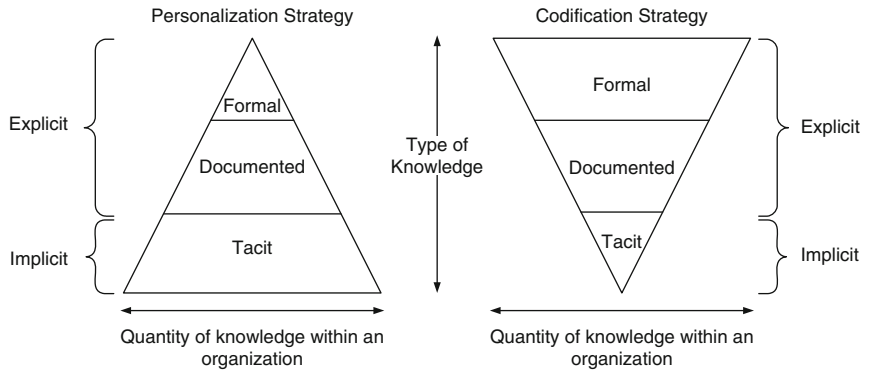

Fig. 6.1 Pyramid of knowledge types and the associated knowledge management strategies [159]

personalization strategy without any codification. However we will present some technologies for personalization in Sect. 6.5. Some of the technologies are mature and can be used off-the-shelf while others are still applied in an experimental setting and are presented here as a future research challenge.

6.2 Use Cases of AK Management

In this section, we describe use cases for AKM tooling to present the potential set of tool features in this domain and also set the stage for presenting existing tools and technologies (discussed in Sects. 6.3–6.5). The use cases define the requirements for developing an AKM tool, i.e. who would use it (actors), to do what (use cases)? We came up with this use case model by surveying a series of papers [10, 113, 192, 326], which provide at least some kind of usage of AK (requirements, services, functions, use cases, etc.). We formed a superset of use cases by selecting, merging, and generalizing from the use cases of the individual approaches. Some of them came from interviews with practicing architects, while others originate from researchers' experience. Furthermore some use cases have been implemented in tools, while others remained in the "wish-list" of researchers.

6.2.1 Actors

Who would use the AKM tool?

- *Architects* designing a system (or a subsystem of a large system) by making decisions. They keep the tacit AK in mind or transform it from tacit to documented or formalized knowledge [326].
- *Reviewers* involved in judging the quality or progress of an architecture [192].
- *Requirements engineers* who view AK from the problem space [192]. *Developers* involved in the implementation of the architecture design and decisions [326].
- *Maintainers* who evolve or maintain the system and need to understand the correlation between the decisions they take and existing, relevant AK [326].
- *Users* of the AKM tool are the entire set of system stakeholders [192]. All the actors mentioned above are specialized actors of *User*.

6.2.2 Use Cases

We present the use cases (UC) by grouping them into four distinct categories, as illustrated in Fig. 6.2: Actors either *consume* AK by using it for specific purposes, or *produce* AK by creating new or modifying existing AK [195]; *knowledge management* concerns general low-level functionality to manage AK data; and *intelligent*

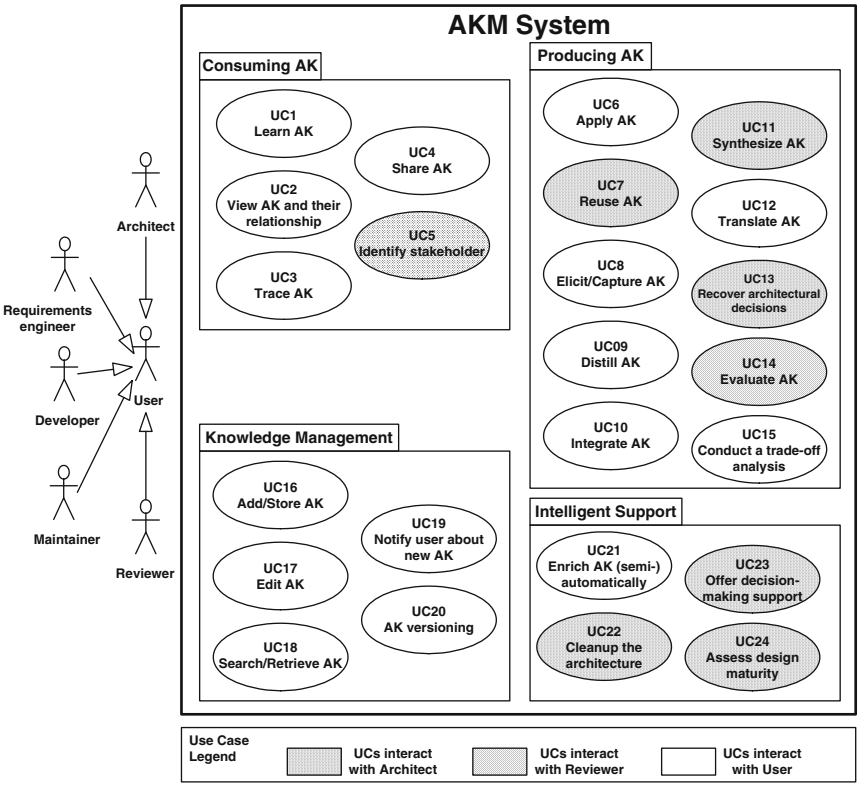

Fig. 6.2 Panorama of the AKM use case model

support concerns automating AKM tasks within the architecting process that require either rigor or intelligence.

We do not claim that the list of use cases is exhaustive, but they do capture all use cases in the surveyed literature. Some use cases are kept unchanged from the original source (e.g. *Assess design maturity* [326]) while others have been merged (e.g. *Reuse AK* [326] includes *Clone AK* [192] and *Stimulate reuse of best practices* [113]), or generalized (e.g. *Identify stakeholders* is generalized from *Identify the subversive stakeholder* [192] and *Identify affected stakeholders on change* [326]). These use cases, together with their included and specialized use cases, are discussed within the presentation of the AKM tools in Sects. 6.3 and 6.4. In this section, we very briefly discuss each use case, and refer to the original sources for more information. It is noted that the actors are explicitly specified only for the use cases whose actor is the *Architect* or the *Reviewer* and not the generic *User*. Also we consider the generalization relationship between use cases as in [40].

Consuming AK

- UC1, *Learn AK* [313]: learn and comprehend the AK, e.g. understand the rationale of a design decision.
- UC2, *View AK and their relationships* [196]: view both AK elements and relationships e.g. the architectural decisions made and the relationships between these decisions.
- UC3, *Trace AK* [12]: trace between various AK elements, e.g. design decisions, rationale, and design.
- UC4, *Share AK* [313]: share knowledge with one or more actors of the system.
- UC5, *Identify stakeholder* [192, 326]: the architect identifies a stakeholder according to certain criteria, e.g. who has the most "weight" on the architectural decisions.

Producing AK

- UC6, *Apply general AK* [313]: use application-independent AK, e.g. apply architecture patterns to solve the problems at hand.
- UC7, *Reuse AK* [326]: the architect reuses AK in another project context, e.g. reusing architectural design decisions from an old to a new project.
- UC8, *Elicit/Capture AK* [10, 196]: elicit and capture AK from various resources, e.g. individuals, teams, or documents.
- UC9, *Distill AK* [313]: distill specific knowledge from a system into general knowledge (e.g. architecture pattern) that can be reused in future systems.
- UC10, *Integrate AK* [313]: integrate different types of information into concrete AK, e.g. integrate stakeholder requirements, system context, and technology constraints into system requirements.
- UC11, *Synthesize AK* [313]: the architect applies the design decisions and produces the system design (e.g. components and connectors).
- UC12, *Translate AK* [208]: translate the formal AK based on a given AK domain model into another domain model to facilitate reuse.
- UC13, *Recover architectural decisions* [326]: the architect reconstructs decisions with their associated rationale from an existing or 3rd party system.
- UC14, *Evaluate AK* [313, 326]: the reviewer performs a critical evaluation of the AK, e.g. to make sure that requirements have been satisfied in the architecture design.
- UC15, *Conduct a trade-off analysis* [326]: analyze the architecture by trading off different quality attributes.

Knowledge Management

- UC16, *Add/Store AK* [192]: add and store elements of AK into the knowledge repository.
- UC17, *Edit AK* [10]: modify or delete AK elements in the repository.
- UC18, *Search/Retrieve AK* [196, 313]: search through the existing AK using certain criteria (e.g. keywords and categories, etc.).

- UC19, *Notify user about AK changes* [113]: subscribe to specific AK elements, and subsequently get notified about changes to them.
- UC20, *AK versioning* [192]: create and manage different versions of the AK.

Intelligent Support

- UC21, *Enrich AK (semi-) automatically* [196]: generate AK content proactively, e.g. automatically distilling and interpreting AK from text without the users' intervention.
- UC22, *Cleanup the architecture* [326]: the architect makes sure that all the dependencies of removed AK (e.g. the consequences of an architectural decision) have been removed as well.
- UC23, *Offer decision-making support* [113, 196]: provide automated support for the Architect in the process of making decisions, e.g. through well-founded advices and guidelines.
- UC24, *Assess design maturity* [326]: the architect evaluates when the architecture can be considered as finished, complete, and consistent, e.g. verify whether a system conforming to the architecture can be made or bought.

6.3 Tool Support for Codification

In this section, we present the AKM tools that support the codification strategy by discussing for each one: a brief introduction, how they support the use cases listed in Sect. 6.2.2 (full or partial support) and their special focus (e.g. architecture design support, evaluation support, etc.). The order of presenting the tools is organized according to the type of knowledge they support: the SEI-ADWiki supports documented AKM; ADkwik and ADDSS support both documented and formal AKM; and the rest support formal AKM.

6.3.1 SEI-ADWiki

This tool is a wiki-based collaborative environment for creating architecture documentation and is used by students in the Carnegie Mellon University Master of Software Engineering program [26]. The tool is not named, so we call it SEI-ADWiki for ease of reference within this chapter.

Supported Use Cases

6.3.1.1 *View AK and their relationships* (UC2): users can view the content of architecture documents and relationships within that content (e.g. mapping between architecture views) through a navigation bar.

6.3.1.2 *Trace AK* (UC3): users can create traceability between architectural artifacts (documented AK represented in wiki pages) through hyperlinks.

6.3.1.3 *Share AK* (UC4): users can share the content of architecture documents with other users after saving the content in wiki pages.
6.3.1.4 *Add/Store AK* (UC16): users can add and save architectural artifacts in wiki pages.
6.3.1.5 *Edit AK* (UC17): users can change and remove the content of architecture document in wiki pages.
6.3.1.6 *Search/Retrieve AK* (UC18): users can use the search box on every wiki page to search the entire wiki by keywords.
6.3.1.7 *Notify user about the AK changes* (UC19): users can opt to receive notification (e.g. by email) when a monitored wiki page (documented AK) is modified by other users or when a new wiki page is created.
6.3.1.8 *AK versioning* (UC20): users can create different versions for the architecture artifacts represented in wiki pages through the versioning function of wiki.
6.3.1.9 *View the change history of AK* (includes UC20): users can view the change history based on the versioned AK. The change history documents which part of the architecture has been added or modified since the last review. AK versioning produces versioned AK for viewing the AK change history, therefore this UC includes UC20.

Special Focus
SEI-ADWiki is able to create and maintain architecture documentation in a dynamic and collaborative way. The basic form of knowledge in SEI-ADWiki is the wiki page (documented AK), which provides a combination of editing and version management tools with the advantage of open access.

6.3.2 ADkwik

ADkwik[1] (Architectural Decision Knowledge Wiki) is a Web 2.0 system which supports the collaborative decision-making work of software architects [291]. Similarly to other wikis, the users (team members) only need a Web browser to work with the system. But still ADkwik is different from a plain standard wiki which is explained in details below. An elaborate application of ADkwik is discussed in Chap. 12.

Supported Use Cases

6.3.2.1 *View AK* (UC2): users can use the navigation bar to view the AK in a dynamic wiki page (e.g. all dependencies to an architectural decision) using Web 2.0 context-aware mashup techniques [17].
6.3.2.2 *Trace AK* (UC3): users can create and manage dependency relationships based on the SOAD architectural decision model [346].

[1] www.alphaworks.ibm.com/tech/adkwik.

6.3.2.3 *Share AK* (UC4): users can share AK across project boundaries with other users. Special attention is given to architecture decisions, which are codified in formal AK and shared according to the SOAD model.
6.3.2.4 *Reuse AK* (UC7): users can reuse the AK about enterprise application architectures contained in the architectural decision repository.
6.3.2.5 *Harvest AK* (*generalized* UC from UC8 and UC9): users can update the AK with new decisions, experiences, patterns and rationale gathered by both successful and failed projects. This UC concerns eliciting/capturing AK (e.g. decisions) and distilling AK (e.g. patterns) and, therefore it is a generalized UC from UC8 and UC9.
6.3.2.6 *Search/Retrieve AK* (UC18): users can search/retrieve the AK from the tagged wiki pages.
6.3.2.7 *AK versioning* (UC20): users can manage different versions of the AK by tracing the wiki changes at the page level.
6.3.2.8 *Offer decision-making support* (UC23): users can find and reuse appropriate decisions in the architectural decision repository.

Special Focus
The main difference of ADkwik from other wikis is that ADkwik is an application wiki as opposed to a plain standard wiki. It is supported by relational database underneath whose tables are structured based on the SOAD domain model [346], while standard wikis also have databases, but the tables are wiki pages. The AK in ADkwik is also structured according to SOAD model to enable formal AK sharing between stakeholders and projects. Consequently ADkwik combines the open access of wikis and formal knowledge based on the underneath domain model to provide efficient AK sharing and management.

6.3.3 ADDSS

ADDSS[2] (architecture design decision support system) is a Web-based tool for storing, managing and documenting architectural design decisions taken during the architecting process and providing traceability between requirements and architectures through the decisions [67].

Supported Use Cases

6.3.3.1 *Learn AK* (UC1): users can understand the architectural decisions by viewing and replaying the evolution of decisions over time.
6.3.3.2 *View AK and their relationships* (UC2): users can easily view the AK and their relationships presented in Web pages and structured according to specific templates.

[2] http://triana.escet.urjc.es/ADDSS.

6.3.3.3 *Trace AK* (UC3): users can trace architectural decisions to other elements (e.g. Architecture, Stakeholder) based on the ADDSS architectural decision domain model.

6.3.3.4 *Share AK* (UC4): users can interact and share AK through groupware support in order to check and solve their conflicts. Another solution to share AK in ADDSS is to generate standard documents (e.g. PDF) with all the architectural information, and send it to related stakeholders.

6.3.3.5 *Elicit/Capture AK* (UC8): users can capture the architectural design decisions using a template that consists of mandatory and optional attributes.

6.3.3.6 *Chronological view of AK* (*included* in UC2): users can view the architectural decisions in a chronological order, to better understand the decision making process. This is a special case of viewing AK and their relationships and, therefore it is an included UC of UC2.

6.3.3.7 *Add/Store AK* (UC16): users can add/store architectural design decisions or design patterns and architectural styles. Decisions can be described using free text, and patterns/styles can be described using graphical and textual description.

6.3.3.8 *Search/Retrieve AK* (UC18): users can query the system about related requirements, decisions and architectures.

6.3.3.9 *Offer decision-making support* (UC23): users can make design decisions by selecting well-known design patterns and architectural styles.

Special Focus

ADDSS uses a flexible approach based on a set of mandatory and optional attributes for characterizing architectural design decisions. Hence, ADDSS provides a customizable codification strategy that makes capturing AK more flexible. ADDSS focuses on the evolution of AK by capturing both architecture designs and architectural design decisions following an iterative process, and visualizing this evolution over time. ADDSS also stresses on applying general knowledge (e.g. architectural patterns).

6.3.4 Archium

The Archium[3] [161, 163] aims at providing traceability among a wide range of AK concepts such as requirements, decisions, architecture descriptions, and implementation. The tool facilitates the maintenance of this AK (e.g. resolve conflicts and synchronize various parts) during the life cycle of a system. All these AK concepts can be expressed in a single language: the Archium language.

Supported Use Cases

6.3.4.1 *Trace AK* (UC3): users can trace architectural decisions to requirements, architectural models and implementation in the Archium language.

[3] www.archium.net.

6.3.4.2 *Distill patterns of architectural decision dependencies* (*specialized* of UC9): users can discover general patterns of dependencies between architectural decisions by viewing their functional dependencies. The patterns of architectural decision dependencies are a type of AK, so this is a specialized UC of UC9.

6.3.4.3 *Add/Store AK* (UC16): users can add/store architectural decisions specified in the Archium language into the repository.

6.3.4.4 *Retrieve AK* (*included* in UC18): users can manually select any component or connector and retrieve the relevant architectural decisions.

6.3.4.5 *Check superfluous architectural decisions* (*included* in UC22): users can identify one class of superfluous decisions in Archium: the unnecessary decisions that have no effect on the architecture. Checking superfluous architectural decisions is a prerequisite for cleaning up the architecture and, therefore this UC is included in UC22. The same reason applies for the next UC.

6.3.4.6 *Get consequences of an architectural decision* (*included* in UC22): users can get the consequence of a decision by viewing the dependency graph of architectural decisions.

6.3.4.7 *Check for consistency of architectural decisions* (*included* in UC24): users can employ the Archium compiler and run-time environment supporting the Archium language to check different types of consistency between architectural decisions, design and implementation. For example, they can check whether dependencies such as *refines* or *dependsOn* of an architectural decision are satisfied with other decisions. Checking consistency of architectural decisions is part of accessing design maturity and therefore this is an included UC of UC24. The same reason applies for the UCs 6.3.4.8–6.3.4.10.

6.3.4.8 *Validate architectural decisions against requirements* (*included* in UC24): users can check whether all requirements are addressed in one or more architectural decisions based on the Archium language.

6.3.4.9 *Check implementation against architectural decisions* (*included* in UC24): the Archium compiler generates code by transforming the architectural elements (e.g. components, connectors) into Java classes. During the process, the compiler also analyzes and verifies whether the transformed Java classes comply with the architectural decisions.

6.3.4.10 *Check for completeness* (*included* in UC24): users can check whether any elements (e.g. motivations, causes, and problems) of an architectural decision are missing.

Special Focus

Visualization and traceability of architectural decisions are the core features of Archium, which are essential for better understanding the architecture design. Archium provides a pragmatic approach to using architectural decisions in architecting: the decisions are bidirectionally linked with the system implementation through transformations, which are transparent to user.

6.3.5 AREL

AREL[4] (Architecture Rationale and Elements Linkage) is a UML-based tool that aims in creating and documenting architectural design with a focus on architectural decisions and design rationale [316]. Three types of AK are captured in AREL: *design concerns, design decisions* and *design outcomes*, which are all represented in UML. An extensive example of the use of AREL in design reasoning is provided in Chap. 9.

Supported Use Cases

6.3.5.1 *Learn AK* (UC1): users can understand the design outcome with its associated *design rationale* (*concern* and *decision*) based on the AREL causal model.

6.3.5.2 *View AK* (UC2): users can view the AK elements and relationships in UML diagrams.

6.3.5.3 *Trace AK* (UC3): users can trace *design concerns* and *design outcomes* to *design decisions* using the UML dependency relationship.

6.3.5.4 *Identify AK change impacts* (*included* in UC14): users can identify all the *design decisions* and other AK elements, that are directly or indirectly impacted when AK is modified, based on the AREL causal model. This UC provides information for evaluating AK, e.g. evaluate the impact of AK change, so this is an included UC of UC14.

6.3.5.5 *Elicit/Capture AK* (UC8): users can capture AK during the architecting process using a UML modeling tool. They can also elicit AK from text-based requirement specifications using UML models.

6.3.5.6 *Synthesize AK* (UC11): users can implement design decisions into the system design in UML diagrams, based on the AREL domain model.

6.3.5.7 *Conduct a trade-off analysis* (UC15): cross-cutting concerns often require trade-off analysis at multiple decision points, and users can conduct such an analysis by tracing between *design concerns* and *design outcomes* that implement the cross-cutting concerns (especially the non-functional requirements).

6.3.5.8 *Add/Store AK* (UC16): users can save the elicited/captured AK in UML models.

6.3.5.9 *Edit AK* (UC17): users can edit the AK through the corresponding UML models.

6.3.5.10 *Search/Retrieve AK* (UC18): users can use the search and retrieval functions provided by the UML modeling to find AK elements within the UML models.

6.3.5.11 *Detect architecture design conflicts* (*included* in UC24): users can detect the design conflicts by looking at the missing links (design gaps) between *design concerns* and *design outcomes* using the AREL causal model. This

[4] www.ict.swin.edu.au/personal/atang/AREL-Tool.zip.

UC can be used for accessing design maturity and, therefore it is included in UC24.

Special Focus
AREL represents various AK elements using UML profiles, thus integrate AKM into a UML modeling tool (e.g. Enterprise Architect). This enables the architect to record the design decisions as part of the architecture design. AREL focuses on linking the problem space (*design concerns*) to the solution space (*design outcomes*) through design decisions in a uniform way.

6.3.6 Knowledge Architect

The Knowledge Architect (KA) is a tool suite for capturing, using, translating, sharing and managing AK. It is based on a common AK repository accessed by different clients (Document Knowledge Client, Excel and Python Plug-in, Knowledge Explorer and Knowledge Translator) [208]. The tool suite makes extensive use of technologies developed for the Semantic Web to allow for formal AK management. The Knowledge Architect is one outcome of the GRIFFIN project, discussed in Chap. 8.

Supported Use Cases

6.3.6.1 *View AK and their relationship* (UC2): users can view the AK entities and their relationships in the Knowledge Explorer.
6.3.6.2 *Trace AK* (UC3): users can create traceability between AK entities using the different client tools.
6.3.6.3 *Share AK* (UC4): users can share AK entities with other users by storing it centrally in the Knowledge Repository and accessing it using the various client tools.
6.3.6.1 *Elicit/Capture AK* (UC8): users can elicit/capture AK by annotating architecture documents and models using the KA client tools.
6.3.6.5 *Integrate AK* (UC10): users can integrate various types of AK (from requirements to design artifacts) into the Knowledge Repository based on a common domain model.
6.3.6.6 *Translate AK* (UC12): users can perform automatic translation based on different AK domain models through the Knowledge Translator [207].
6.3.6.7 *Add/Store AK* (UC16): users can save the captured (annotated) AK entities into the Knowledge Repository through the client tools.
6.3.6.8 *Edit AK* (UC17): users can edit the AK entities through the client tools.
6.3.6.9 *Search/Retrieve AK* (UC18): users can query the AK entities and their relationships in the Knowledge Repository through its Query Engine, using the RDF query language.
6.3.6.10 *Check completeness of AK* (*included* in UC24): users can check the completeness of AK in a document (e.g. whether a *Decision Topic* has been addressed by at least one *Alternative*) through the Document Knowledge

Client. Checking completeness of AK is part of accessing design maturity, so this is an included UC of UC24.

Special Focus
The Knowledge Architect focuses on capturing AK through annotating information in different sources (e.g. Word, Excel documents), and sharing it in a central repository. The tools suite also focuses on traceability management and intelligent support, as AK entities and relationships are semantically specified in OWL [37].

6.3.7 SEURAT

SEURAT[5] (Software Engineering Using RATionale system) is an Eclipse plug-in that is targeted to rationale knowledge management in an integrated development environment (IDE), from requirements to design and finally to source code [60][62]. The concept of rationale knowledge in SEURAT is composed of design decisions, alternative solutions considered, and the reasons (arguments for each solution) behind the final decisions.

Supported Use Cases

6.3.7.1 *Learn AK* (UC1): users can understand rationale knowledge and all its parts. It is represented with a formal argument ontology (for details on the argument ontology see [59]), which semantically assists the understanding of the rationale knowledge.

6.3.7.2 *View AK* (UC2): users can view rationale knowledge within the Eclipse environment in a hierarchical view – from list of decisions to alternative solutions and finally to the "arguments" for or against each solution. Users can also view rationale knowledge through the rationale hierarchy report (in the same layout as in the hierarchical view) and the rationale traceability matrix report generated by the tool.

6.3.7.3 *Trace AK* (UC3): users can trace the rationale knowledge (e.g. "*how do we compare dates?*") to source code (e.g. function *compareDates()*) directly in the Eclipse environment using bookmarks. Users can also trace requirements to the decisions made and captured in the rationale.

6.3.7.4 *Elicit/Capture AK* (UC8): users can capture rationale knowledge during source code development. They can also import rationale knowledge from Word documents where text has been annotated as rationale.

6.3.7.5 *Decision evaluation and impact assessment* (*included* in UC14): users can evaluate decisions by calculating the "support score" for each alternative solution based on the arguments for and against it. Users can also disable some requirements when stakeholders change their mind about them, and see which decisions may require re-examination due to the impact

[5] www.users.muohio.edu/burgeje/SEURAT/Downloads.html.

assessment of requirements. This UC aims at evaluating design decisions, a type of AK, so this is an included UC of UC14.

6.3.7.6 *Conduct a trade-off analysis* (UC15): users can conduct trade-off analysis of a decision based on the "background knowledge" (e.g. "*A more flexible solution costs more to develop*") of this decision which is explicitly recorded in the rationale.

6.3.7.7 *Add/Store AK (UC16)*: users can add rationale knowledge using an editing interface integrated in Eclipse and store it in a relational database.

6.3.7.8 *Edit AK* (UC17): users can edit the text of rationale knowledge using the editing interface.

6.3.7.9 *Search/Retrieve AK* (UC18): users can search/retrieve rationale knowledge elements through keyword-based search, including requirements, decisions, alternatives and arguments.

6.3.7.10 *Offer decision-making support* (UC23): users can get decision-making support using the "support scores" for each alternative solution.

6.3.7.11 *Check for completeness and consistency of rationale knowledge (*included *in UC24)*: users can detect the incompleteness and inconsistency of the rationale knowledge through inferencing based on the argument ontology, e.g. for completeness, checks are made to ensure that there are alternatives proposed for each decision. This UC can be used for accessing design maturity and, therefore it is included in UC24.

Special Focus

SEURAT is not specifically used for the management of AK but for rationale knowledge. However, in a broad sense, rationale knowledge about architecture design (e.g. arguments linked from requirements to alternative design solutions) is an important part of AK. In addition, SEURAT mainly focuses on the application of rationale knowledge supporting software maintenance [61].

6.4 Tool Support for the Hybrid Strategy

In this section, we present the AKM tools that support the hybrid strategy in the same structure as in Sect. 6.3.

6.4.1 EAGLE

EAGLE [114, 113] is an AK sharing portal that implements best practices from knowledge management for improving AK sharing. The main features of EAGLE include integrated support for both codified and personalized AK, support for stakeholder-specific content, and AK subscription and notification. EAGLE is a result of the GRIFFIN project, discussed in Chap. 8.

Supported Use Cases

6.4.1.1 *Share AK* (UC4): users can share AK in both personalized (e.g. news, events and experience with colleagues) and codified (e.g. best practices, documents) formats.

6.4.1.2 *Find a colleague based on expertise or competence* (*included* in UC4): users can find the right person, whose personal knowledge may match a specific AK request. This UC provides information for personalized AK sharing, so it is an included UC of UC4.

6.4.1.3 *Overview of personal information of colleagues* (*included* in UC4): users can get an overview of "who knows what" and "who is doing what" among their colleagues. This UC also provides information for personalized AK sharing, so it is an included UC of UC4.

6.4.1.4 *Add/Store best practices* (*specialized* of UC16): users can add best practices to a repository (codified AK) for reuse and decision making support. Best practices are a special type of AK, so this is a specialized UC of UC16.

6.4.1.5 *Add/Store architecture document* (*specialized* of UC16): users can add architecture documents to a repository according to various AK categories. Similarly to the UC in 5.1.4, this is also a specialized UC of UC16.

6.4.1.6 *Search/Retrieve AK* (UC18): users can access generic documentation (different types of company documents) by document title, keywords or categories, and also search for project-specific AK documentation.

6.4.1.7 *Search/Retrieve related AK* (*included* in UC18): users can access external information sources to find related AK, such as white papers (codified AK), seminars and trainings (personalized AK) or other corporate communication, e.g. discussion board (personalized AK). Related AK is a special kind of AK, so this is a specialized UC of UC18.

6.4.1.8 *Notify user about new AK* (*specialized* of UC19): user can stay up-to-date about new AK through subscription and notification mechanisms. New AK is a kind of AK change and, therefore this is a specialized UC of UC19.

6.4.1.9 *Offer decision-making support* (UC23): Users can get intelligent support by answering a questionnaire during the decision-making process, and automatically receiving a number of architectural guidelines that match their answers.

6.4.1.10 *Overview of project stakeholders* (*included* in UC24): users can have an overview about project stakeholders, e.g. contact information and expertise area. They can subsequently request all the involved stakeholders to access the design maturity and, therefore this is an included UC of UC24.

Special Focus

EAGLE focuses on stakeholder collaboration during the architecting process, by enabling them to connect to colleagues or other involved stakeholders by retrieving "who is doing what" and "who knows what". In addition, codified AK in a document repository or best practice repository can also be easily accessed using advanced search mechanisms.

6.4.2 PAKME

PAKME[6] (Process-based Architecture Knowledge Management Environment) is a Web-based tool aimed at providing knowledge management support for the software architecture process. PAKME supports both codification and personalization as it not only provides access to AK but also identifies the knowledge source [8, 12].

Supported Use Cases

6.4.2.1 *View AK and their relationships* (UC2): users can view AK elements (e.g. architectural patterns) in template-driven Web pages, and their relationships through hyperlinks.
6.4.2.2 *Trace AK* (UC3): users can trace AK using hyperlinks and relationship types (e.g. *constrain* or *conflictWith* relationships between architectural design decisions) defined in PAKME.
6.4.2.3 *Share AK* (UC4): users can share the AK stored in the PAKME repository through the Web user interface.
6.4.2.4 *Apply general AK* (UC6): users can apply general AK (e.g. patterns, general scenarios) to design a suitable architecture for a new application.
6.4.2.5 *Reuse AK* (UC7): users can reuse alternative design solutions in 4 steps (searching, retrieving, reviewing and integrating).
6.4.2.6 *Elicit/Capture AK* (UC8): users can use various Web forms based on templates (e.g. architectural decision and pattern templates) to elicit, structure and capture AK before storing it into the repository.
6.4.2.7 *Add/Store AK* (UC16): users can use various Web forms to enter generic or project-specific AK into the repository, including the knowledge producer information.
6.4.2.8 *Edit AK* (UC17): users can modify and delete the AK stored in the repository through the Web user interface.
6.4.2.9 *Search/Retrieve AK* (UC18): users can search/retrieve AK elements through keyword-based search, advanced search and navigation-based search. For personalization purposes the source of AK (e.g. knowledge producer) can also be retrieved.

Special Focus
PAKME focuses on various collaborative AK management features for geographically distributed stakeholders involved in the architecture process by managing and sharing codified AK (pattern, decision, etc.) and personalized AK (contact management, online collaboration, etc).

6.5 Technologies

Some of the AKM tools described in the previous sections were not built from scratch but made use of various technologies. Such technologies are generic and can be employed to support the AKM use cases presented in Sect. 6.2. In this section, we

[6] http://193.1.97.13:8080/.

present a number of these technologies to demonstrate their value for AKM tools, and to assist tool vendors in selecting the appropriate ones for their own needs. The order of presenting the technologies is organized according to the type of knowledge they support: Web portal, blog and wiki support the hybrid strategy, voting and ranking support the personalization strategy, and the rest support the codification strategy.

6.5.1 Web Portal

A Web portal [131, 319] is a Web site that provides integrated modules, like hosted databases, yellow pages, discussion boards, news push, document management, email and more. Web portals automatically personalize the content generated from these modules to provide a personalized experience to users. The yellow pages module can record the expertise area, involved projects and contact information of all the architects in an organization, thus providing support for personalized *AK sharing* (UC4). Emails, news push and discussion boards provide communication support for *AK sharing* (UC4) through a collaboration space among users. News push also supports *AK changes notification* (UC19) when personalized information is changed (e.g. personnel movement).

This technology is also useful for codified AK management. The hosted central databases and client/server architecture can facilitate *AK sharing* (UC4), and Web forms can be used for *tracing* (UC3), *eliciting/capturing* (UC8), *adding/storing* (UC16), and *editing* (UC17) *AK*.

6.5.2 Blog and Wiki

Different from yellow pages, blogs and wikis are both editable Web pages by individual users who are distributed over the network. Blogs are for single users, and wikis are designed to enable anyone to contribute or modify content, so they both support personalized *AK sharing* (UC4). For example individual users can provide up-to-date and more reliable personal information, such as their expertise area and personal knowledge.

As a collaborative Web editing system, wikis also support codified AK management for both documented and formal AK. We classify the wikis as general wikis for documented AKM (e.g. SEI-ADWiki) and semantic wikis for formal AKM (e.g. ADkwik). Both types of wikis can support the following AKM use cases: *AK viewing* (UC2) and *AK traceability* (UC3), *adding/storing AK* (UC16), *editing AK* (UC17), *searching/retrieving AK* (UC18), *user notification about AK changes* (UC19), and *AK sharing* (UC4). Some practical experience of applying wikis to support AK sharing can also be found in [116].

Generic wikis simply document AK in the wiki pages. On the other hand, semantic wikis provide semantic support through formal models (e.g. semantic annotation and semantic query of AK). In addition, wikis have also been used in requirements engineering to support requirements knowledge management in a codification strategy, e.g. for documented requirements [91] and formal requirements knowledge [212].

6.5.3 Voting and Ranking

Voting and ranking is a method to evaluate and rank the characteristics (e.g. credibility and reliability, etc.) of objects or people by different users in an online community. It has been widely applied in many social networking systems (e.g. LinkedIn) and C2C business platforms (e.g. eBay) for the evaluation and ranking of personal information.

The personal information recorded in Web portals, wikis and yellow pages has unavoidably personal and subjective bias (e.g. the expertise of an architect). Using the voting and ranking mechanism can partially mitigate this problem, and provide more credible personal information. For example ranking the expertise of different stakeholders on a technology platform by other members of an organization helps to create reliable "who knows what" information, and thus efficient personalized *AK sharing* (UC4).

6.5.4 Natural Language Processing

Natural language processing (NLP) is concerned with the understanding of human natural languages by computers. Since documentation in natural language is dominant in AK resources (most documented AK is in natural language), it is beneficial to introduce NLP techniques in AKM tools.

Several AKM use cases have been supported by NLP techniques. The LSA (Latent Semantic Analysis) technique has been used to *elicit/capture AK* (UC8) semi-automatically [45]. Text mining techniques have been used to *enrich AK* (UC21) [114], and *offer decision-making support* (UC23) [196].

6.5.5 Ontologies

Ontologies are formal structures supporting knowledge representation, management, sharing and reusing [120], and have been widely used in various fields, such as the Semantic Web. They represent explicitly the semantics of structured and

semi-structured information and so enable sophisticated automatic support for acquiring, maintaining and accessing information [90]. Formal AK, as a kind of formal knowledge, can be represented by ontology models (see for example [190]). Various ontology techniques have been explored in AKM tools, including ontology modeling, ontology database, ontology mapping, and ontology-based inferencing. These techniques will be elaborated in the following paragraphs.

Ontology modeling can be used to describe domain concepts and their relationships. In this respect, AK ontology models are composed of AK domain concepts (e.g. *Design Decision*, *Alternative*, and *Risk*) and relationships (e.g. *addressedBy* and *containedIn*). Related standards on ontology specification to specify the ontology modeling results have been defined by the W3C, e.g. RDF [184] and OWL [37], with ontology modeling tools support, e.g. Protégé[7]. Combined with ontology models, an AKM tool can support the following use cases of formal AK management: UC1 (*Learn AK*) - the ontology concepts and relationships can help users understand the meaning of AK; UC2 (*View AK and their relationships*), UC3 (*Trace AK*) and UC22 (*Clean up architecture*) - these use cases are supported by using the semantic relationships defined between AK concepts.

Ontology databases store data in ontological data models. For example the RDF store Sesame [53] stores data in the RDF triple format. Ontology databases provide semantic querying using their specific query language, e.g. SPARQL [258] of the W3C, SeRQL for Sesame. For example, one can query the ontology database by posing question like "*Tell me all the alternative solutions addressed to decision topic 'the control method over the data processing pipelines' which are not in conflict with each other*". The following use cases can be supported by ontology databases in formal AK management tools: *Add/Store AK* (UC16) in ontological data models, *Search/Retrieve AK* (UC18) by query languages, and *Share AK* (UC4) after getting the query results.

Ontology mapping is an activity to semantically relate two ontologies [112]. It provides a semantic translation between heterogeneous ontologies and therefore enables knowledge sharing in a semantically sound manner [169]. This is essential for sharing AK that originates from different organizations and is based on different AK ontologies. UC4 (*Share AK*) and UC12 (*Translate AK*) can be supported by ontology mapping techniques (see e.g. the Knowledge Translator [208]).

Ontology-based inferencing concerns retrieving knowledge and creating deductive knowledge based on ontology models with logic-based reasoning [333]. In AKM tools, the inferencer can be mostly used to automatically infer the relationships that exist between the formal AK entities, e.g. an inverse relationship between AK elements (for traceability) or a mapping relationship between elements from different AK domain models (for translation). The inferencer can support the following use cases: *Search/Retrieve AK* (UC18) e.g. by the SeRQL query language of OWLIM inference engine [181], *Check for completeness* (included in UC24) e.g. by the KA client tools [208], and *Translate AK* (UC12) e.g. by the Knowledge Translator [208].

[7] http://protege.stanford.edu/.

6.5.6 Plug-in

A plug-in consists of a program that connects and interacts with a host system (e.g. a Web browser or an email client) to provide a specific function on demand. This technology is quite beneficial for promoting AK usage through tools that architects have being working and are familiar with. Typical tools that architects use through the architecting process include word processors for architecture documentation, spreadsheets for quantitative architecture evaluation and UML modelers for architecture design.

Examples of the tool plug-in technology include the KA Document Knowledge Client (Word plug-in) and AREL (UML modeling tool plug-in). Both plug-ins support the following AKM use cases: *Learn AK* (UC1), *View AK* (UC2), *Trace AK* (UC3), *Elicit/Capture AK* (UC8), *Synthesize AK* (UC11), *Add/Store AK* (UC16) and *Edit AK* (UC17).

6.5.7 Version Management

This technology concerns the management of multiple revisions of the same unit of information. The versioning function of wikis, SVN (Subversion) and CVS (Concurrent Versions System) are typical examples of this technology. Wikis can track the version changes at the page level. For every page, it is easy to look up earlier versions, display the differences between two versions, and revert to an older version. SVN and CVS provide similar functions for the version management of files which can be used to record documented AK (e.g. Word documents) and also formal AK (e.g. RDF files). AK evolves rapidly during the architecting process, and effective version management of AK can support directly *AK versioning* (UC20) and indirectly *viewing the change history of AK* (includes UC20).

6.5.8 Web 2.0

Web 2.0 aims to enhance creativity, information sharing, collaboration and functionality of the Web. Interesting techniques in Web 2.0 for codified AKM include push and pull mechanisms, tags and context-aware mashups. Push and pull mechanisms (e.g. RSS – Rich Site Summary) can be used to *notify user about AK changes* (UC19) for subscribed users [114]. Tags can be used to *search/retrieve AK* (UC18) for Web pages that have been tagged. Context-aware mashups can be used to *view AK and their relationships* (UC2) e.g. all the inter-dependent elements of an architectural decision can be shown in a dynamic mashup Web page, which combines the AK elements from more than one source into a single integrated Web page [17].

6.6 Summary

For the effective usage of AK in the architecting activities, the AKM tools have been recognized as a great contribution [7]. In this chapter, we provide a survey of current tools and technologies with respect to the AKM strategies they adopt and the use cases they support. We hope to help AKM tool developers in understanding the state-of-the art and practice and get inspired in building their own tools. We expect that depending on their specific needs and organizational context, they will mix and match the appropriate technologies and ideas from existing tools, in order to build customized AKM tools. We are confident that, as more AKM tools are built, more AK will be used in practice and shared among organizations and thus contribute to establishing AKM in the daily organizational practices.

It is noted that the following use cases, identified in Sect. 6.2, have not been fully supported (implemented) by existing AKM tools: UC5 (*Identify stakeholder*), UC11 (*Synthesize AK*), UC13 (*Recover architectural decisions*), UC14 (*Evaluate AK*), UC21 (*Enrich AK (semi-) automatically*), and UC24 (*Assess design maturity*). We regard these use cases as the future challenges, which AKM tool developers can work on to provide more added value to existing AKM. We also believe that some promising technologies can be the key for implementing these use cases, such as NLP for intelligent support (advices and guidelines for making decisions) [196], context-aware text mining for the elicitation of user interests about AK, and ontology inferencing for the enrichment of AK.

Acknowledgements This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge.