Die Grundlage zur Entwicklung der CAFM.*TOOLS* ist die Legal Compliance CAFM Strategie 2.0

Ergebnisse Anwenderkreis: Universitäten, Kliniken, Hochschulen, Kommunen, Städte

Ziel war es die Erfahrungen zum Thema CAFM Software für die Entwicklung einer praxisnahen und geeigneten CAFM Strategie zu berücksichtigen. Dabei sollten die Ursachen und Gründe von Problemen, die bei der Auswahl, Einführung und Nutzung von CAFM Systemen aufgetreten sind ermittelt werden und durch eine geeignete Strategie in Zukunft weitestgehend vermieden werden. Als Ergebnis wurden die Anforderung an das Gesamtsystem (Software, Betriebssystem, Datenbank, Programmiersprache, Hardware) in der CAFM Strategie 2.0 konkretisiert und mit Unterstützung aller Beteiligten seit 2016 Schritt für Schritt in dem Projekt CAFM.TOOLS umgesetzt. Im Ergebnis sind bereits ca. 70% der Anforderungen der Anwender bereits umgesetzt und nutzbar. Freie Software (Open Source) für Universitäten Kliniken Hochschulen als CAFM

Vereinbarungen und Anforderungen Forum: "Anwender für Anwender" (Auszug CAFM Strategie 2.0 Stand 08.2022)

§1) Die Ziele und Anforderungen

Welche Blickrichtungen sollen betrachtet werden und dabei helfen eine realistische und planbare Zukunftsstrategie abzuleiten.
Welche Nutzung steht im Vordergrund Aktuelle CAFM Situation (Was möchten wir verbessern bzw. in Zukunft vermeiden) Wie kann eine zukunftssichere realistisch und planbare Strategie konkret aussehen (Eigenschaften) Welche Rückschlüsse sollten wir mit einem Blick in die Vergangenheit ziehen

§2) Nutzung des Systems und der Daten

Welche Eigenschaften müssen bei der Strategieentwicklung berücksichtigt werden bzw. stehen im Vordergrund.
Der selektive mobile Zugriff (intern, extern) auf die Daten (Anlagendaten, Dokumentation, Historie, Prozesse) soll allen Beteiligten auf einfachste Weise möglich sein (auch Vorort an der Anlage, große Nutzerzahl erwünscht) Kein leeres System, sondern eine schlüsselfertige Lösung mit vorkonfigurierten Standards (Legal Compliance, Betreiberpflichten, Anlagenmerkmale, Dokumentationsvorgaben) Die Tendenz geht zu einer Plattform für hochspezialisierte TOOLS und weg von der Suche nach einer ultimativen Gesamtlösung Keine Redundante Datenhaltung. Daten sollen in anderen Anwendungen genutzt werden können, aber zentral gepflegt werden

§3) Anforderungen an die Verfügbarkeit

Dabei stehen die Nutzung der Daten **und** der gesamten Entwicklungsumgebung im Vordergrund
Der Zugriff und die Verfügbarkeit der CAFM Daten sollen über den gesamten Lebenszyklus der Anlagen bzw. Gebäudes gesichert sein Langfristige Verfügbarkeit und Abwärtskompatibilität (30 Jahre) der eingesetzten Entwicklungsumgebung (Softwarepaket) und Komponenten (Datenbanken, Programmiersprachen, Betriebssysteme, Webserver, Webbrowser, Hardware, Netzwerke, usw.) Keine Abhängigkeit über Lizenzen von einem Hersteller, Lieferanten oder Dienstleister für die Gesamtlösung (Hosting, Programmierung, Datenbank,

Programmiersprache, Betriebssystem, Hardware, CAFM Dienstleister (Daten werden als Geisel eingesetzt) Trennung von Hosting der Software (Anwendung) und dem Hosting der Daten (Dokumente und Datenbanken). Dabei soll es möglich sein, die Software als Software as a Service (SaaS) inkl. Support zu nutzen und die Daten in der Cloud des eigenem Rechenzentrum (stellt nur Festplattenplatz zur Verfügung) abzuspeichern

§4) Anforderung an die Pflege und Weiterentwicklung

Grundlagen und Anforderungen an die funktionelle Weiterentwicklung der Software und gesamten Entwicklungsumgebung
Schnelle, planbare und kostengünstige Weiterentwicklung des Systems durch die Verfügbarkeit vieler Lieferanten für projektunterstützende Dienstleistungen (Programmierung, Beratung, Schulung, Einführung, Hosting) Gemeinsame Pflege und Weiterentwicklung der vorkonfigurierten Standards (Betreiberpflichten, Anlagenmerkmale, Dokumentationsvorgaben) Keine unkalkulierbaren und zusätzliche Kosten für den Erhalt der Daten und der Weiternutzung nach einem Update, bei einer Umstellung auf eine neue Version, einer neuen Hardware oder den Verkauf der Anwendungssoftware usw. Weiterentwicklungen, die von einem Nutzer umgesetzt (bezahlt wurden) stehen allen Nutzern zur Verfügung. Konzentration auf die eigentliche Software und autarke Weiterentwicklung der gesamten Entwicklungsumgebung

§5) Zukunftssichere Technologie

Um den dezentralen mobilen Zugriff und die hohe Zahl an Benutzern zu ermöglichen, bietet der Einsatz der Webtechnologie (Zugang über den vorhanden Webbrowser) gegenüber klassischen Klient Server Lösungen, bei dem der Zugriff auf die Daten nur durch eine vorher installierte Software möglich ist, klare Vorteile.
Empfehlung: Webtechnologie

§6) Transparenter Quelltext

Es gibt einen proprietäre und offenen Ansatz um Software zu entwickeln. Im Gegensatz zu den proprietären haben offene Umgebungen den Vorteil des offenen Quelltextes. Durch die Offenlegung und Dokumentation des gesamten Programmcodes kann im Gegensatz zu einem proprietären (Quelltext nicht lesbar) Ansatz die Funktion der Software transparent nachvollzogen werden und bietet nicht nur für den Datenschutz und Sicherheit erhebliche Vorteile. Das gilt nicht nur für die eigentliche Software, sondern auch für alle eingesetzten Werkzeuge. (Datenbanken, Programmiersprachen, Betriebssysteme, Webserver, Webbrowser)
Empfehlung: Durchgängige offene Software und Entwicklungsumgebung Aktuelles Beispiel mit offenem Quelltext ist die Corona-Warn-App (Telekom und SAP)

§7) Urheberechte beachten

Allein der Umstand, dass der Quelltext lesbar ist, bedeutet nicht automatisch, dass man ihn auch ändern darf. Jeder Quelltext unterliegt einem Urheberrecht. Durch eine Lizenz werden die Art und den Umfang der Nutzung durch den Urheber bestimmt. Bei proprietären Lösungen wird die Möglichkeit der Bearbeitung des Quelltextes durch die Verschlüsselung von vorneherein unterbunden (100% Abhängigkeit). Alternativ gibt es die freie Softwarelizenzen wie z. B. GPLv2 (Open-Source-Software), bei der die Weiterentwicklung ausdrücklich erwünscht ist und somit die Entwicklung nicht mehr an einen Lieferanten gebunden ist. Man wechselt bei Bedarf nicht mehr die Software, sondern die jeweiligen Lieferanten
Empfehlung: Software und Entwicklungsumgebung mit z. B. GPLv2 (Plattform)

§8) Die Entwicklungsumgebung

Die zunehmende Nutzung der Cloudtechnologie (SaaS) steigert wesentlich die Kompatibilität der einzelnen Lösungen untereinander. Dabei hat sich gezeigt, die Lösungen, die eine gleiche Entwicklungsumgebung nutzen, im Gegensatz zu den proprietären Lösungen automatisch kompatibel zueinander sind und Daten untereinander ausgetauscht und genutzt werden können. Die mit Abstand weitverbreitetste Umgebung im Bereich Web ist das sogenannte LAMP Softwarepaket. In der Spezifikation sind die eingesetzten Komponenten definiert. Linux

(Betriebssystem), Apache (Webserver), MySQL (Datenbank) und PHP (Programmiersprache). Der zusätzliche Vorteil ist, dass die gesamte Umgebung über einen offenen und dokumentierten Quelltext verfügt und dieser auch verändert werden kann.
Empfehlung: LAMP Entwicklungsumgebung

§9) Plattform

Im Gegensatz zu dem Einsatz von einer Gesamtlösung (geschlossenen CAFM System von einem Hersteller) wo die Kompatibilität zu anderen Systemen aufwendig nachprogrammiert werden muss, werden auf einer Plattform die Daten über genormte Schnittstellen allen Werkzeugen (auch von verschiedenen Herstellern) zur Verfügung gestellt. Beispiel: Daten aus dem Raumbuch stehen einem Anlagenkataster zur Verortung der Anlage zur Verfügung. Aus dem Raumbuch sieht man welche Anlage in dem Gebäude, Etage oder Raum eingebaut sind.
Empfehlung: Aufbau und Nutzung einer Plattform

§10) Praxisnahe Einführung

In den meisten Fällen klappt die z. B. Instandhaltung auch ohne ein aufwendiges CAFM System. Über viele Jahre haben sich praxisnahe Abläufe entwickelt, die grundsätzlich funktionieren. Die Befürchtung der Mitarbeiter ist es nun diese Abläufe und Werkzeuge aufzugeben zu müssen und nach Einführung eines neuen Systems mit leeren Händen dazustehen. Bei der Entwicklung einer geeigneten praxisnahen Strategie sollte das berücksichtigt werden. Die Anforderung an ein System könnte z. B. konkret mit den Monteuren vor Ort an der Anlage ermittelt werden. In Gesprächen sucht man nach konkreten Ansäten, um die bestehenden Abläufe zu unterstützen. Dabei ist es wichtig jeden Schritt zuerst praxisnah im Tages Geschäft auf ein tatsächliches Nutzen zu überprüfen ohne die bestehenden Abläufe zu behindern. Die Erfahrungen zeigt, dass aus der Vergangenheit sehr viel verbrannte Erde hinterlassen worden ist und das Vertrauen an ein erfolgreiches Projekt manchmal erst aufgebaut werden muss.

§11) Entwicklungsstrategie (Software)

Ziel ist es den Initialaufwand für die Datenbeschaffung, Entwicklung und Einführung von CAFM Funktionalitäten zu verringern. Der tatsächliche erkennbaren Nutzen soll für alle Projektbeteiligen transparent im Vordergrund stehen, bevor die eigentliche Programmierung beginnt bzw. eine Software gekauft werden soll.
Dabei soll zuerst ermittelt werden, ob und wie die Daten die später ausgegeben werden sollen, beschafft und gepflegt werden können In einem zweiten Schritt soll die jeweiligen Daten für einer Testumgebung auszugsweise zusammengetragen werden (Modell) In einem dritten Schritt werden auf der Grundlage des Modells Funktionen ermittelt werden, wie die Daten bearbeitet (Controller) und ausgegeben (View) werden sollen In einem 4 Schritt werden die Funktionen auf der Grundlage des Modells programmiert und getestet
Ein geeignetes Architekturmuster für die Softwareentwicklung ist der Model View Controller. Das Konzept wurde bereits 1997 entwickelt und gilt mittlerweile als Standard für die Entwicklung vieler komplexer Softwaresysteme. Für die Vorgehensweise, bietet die Scrum Methode entscheidende Vorteile. Scrum wurde bereits 1993 als „ Vorgehensmodell“ für die Entwicklung von Software eigesetzt und zählt bis heute immer noch zu den bekanntesten agilen Methoden. Durch seine einfache Struktur und die klar definierten Rollen lassen sich die Scrum-Prinzipien schnell lernen, produktiv einsetzen und so die Vorteile von Agilität schnell ausnutzen.

Zusammenfassung Freie Software (Open Source) für Universitäten Kliniken Hochschulen als CAFM

Einsatz der Webtechnologie Durchgängiger offener Quelltext für die Software und Entwicklungsumgebung Software und Entwicklungsumgebung mit z. B. GPLv2 für Standardanwendungen LAMP Entwicklungsumgebung Nutzung einer offenen Plattform für Anwendungen von verschiedenem Hersteller Konzept der Softwareentwicklung Model View Controller. Vorgehensweise für die Softwareentwicklung Scrum