# Zemprosys Version 2008

**VISUAL PAGE PROGRAMMING TECHNOLOGY**

- Entwicklung von Computerprogrammen ohne Code
- Visuelle Programmierung

# Grundlagen und Strategie

© Zemprosys Group 2008

Verfasser dieses Dokuments: Maurice Ze
maurice@zemprosys.com

Maurice Ze ist Urheber des Softwareprodukts Zemprosys.
Alle Rechte vorbehalten - Vervielfältigungen aller Art - auch auszugsweise - nur mit der schriftlichen Zustimmung von Maurice Ze

# INHALTSVERZEICHNIS

**1. EINLEITUNG....3**

**2. MOTIVATION....4**

2.1. AUSGANGSSITUATION....4
2.2. HEUTIGE ANFORDERUNGEN AN SOFTWARE....4

**3. SOFTWARE-ZIEL....6**

**3.1. PROGRAMMIERUNG DURCH DEN BENUTZER....6**

**4. TECHNOLOGIE....7**

4.1. ÜBERBLICK....7
4.2. FUNKTIONSWEISE....7
4.3. VISUAL PAGE PROGRAMMING TECHNOLOGY....8
4.4. ZEMPROSYS SCRIPT....11
4.5. ZEMPROSYS DESCRIPTOR....15
4.6. ZEMPROSYS NET ENGINE ....15
4.7. PLATTFORMEN....16
4.8. INTEGRATION IN DELPHI....17

**5. ANWENDUNGEN....17**

5.1. ÜBERBLICK....17
5.2. VISUALISIERUNGSSYSTEME....17
5.3. DYNAMISCHE DOKUMENTE....18
5.4. XML APPLIKATIONEN....18
5.5. WORKFLOW APPLIKATIONEN....18
5.6. UML/AML....18
5.7. CBT/WBT....19
5.8. META KOMPONENTEN....19

**6. FORSCHUNG & ENTWICKLUNG....20**

6.1. ÜBERBLICK....20
6.2. ENTWICKLUNGSSTAND....20
6.3. SOFTWAREARCHITEKTUR....21
6.4. ERWEITERUNG DER KOMPONENTENBIBLIOTHEK....22
6.5. SOFTWARE-VERBESSERUNG....24
6.6. EDITIONEN....26
6.7. OPEN ZEMPROSYS....26
6.8. PORTIERUNG DER TECHNOLOGIE....27

**7. ZEMPROSYS INTRO....28**

7.1. ÜBERBLICK....28
7.2. TOOL BESCHREIBUNG....28
7.3. ENTWICKLUNGSPAKET....30

**8. CLIENT/SERVER-MODELL....32**

8.1. ÜBERBLICK....32
8.2. DESIGN....33

# 1. EINLEITUNG

Das vorliegende Dokument soll zur Vermittlung von Zemprosys-Grundlagen dienen. Zemprosys steht für **Ze Maurice Programming System** und repräsentiert eine Softwareplattform zur Erstellung anderer Software auf Basis einer innovativen Produktionsmethode. Die einzigartige Kernleistung von Zemprosys ist es, vollständige Computerprogramme für beliebige Geräte zu erstellen, optimieren, verändern oder konvertieren, ohne dass dafür manuelle Codeschreibung von Softwarespezialisten notwendig ist. Der Bediener arbeitet mit einer graphischen Benutzeroberfläche, während die Software im Hintergrund den optimierten Code für die gewünschten Funktionen erstellt.

Zemprosys ist im ersten Schritt für die Zielgruppe der industriellen Anwender konzipiert, wie z.B. Maschinen-/Roboterbau, Mess-, Steuer- und Anzeigengeräte und jede andere Art von Softwaregesteuerten Hardware. Auch Frontend- und Webentwickler sind eine potenzielle Zielgruppe. Darüber hinaus wird das Produkt dahingehend weiterentwickelt, dass es mit einem Bruchteil der heute notwendigen Ressourcen auch komplexe IT-Strukturen (z.B. für Unternehmen, ERP, MIS ...) im Detail maßgeschneidert erstellt. Der dritte mögliche Einsatzbereich ist der Privatanwender, der eigene Programme auf seinem PC selbst konstruieren kann (z.B. Dokumentenverwaltung oder eine Datenbank).

Durch die Entwicklung von Zemprosys möchten wir die Gedanken der Softwareentwicklung ohne Code maßgeblich prägen und somit die textuelle Programmierung auf Basis einer einzigartigen visuellen Produktionsmethode ersetzen. Unserem Bestreben nach soll jeder Computernutzer in die Lage versetzt werden, eigene Computerprogramme zu entwickeln, ohne unbedingt Kenntnisse in Programmierung bzw. Programmiersprachen haben zu müssen. Die Programmierung durch Einsatz einer bestimmten Programmiersprache spielt dabei eine fast untergeordnete Rolle: Die Programmierung ohne Code steht mehr im Vordergrund.

Die innovative Produktionsmethode VPPT (**Visual Page Programming Technologie**) ist ein zentrales und unverzichtbares Element von Zemprosys, da diese Methode die visuelle Programmierung auf eine einfache Software-Montage reduziert. Die Methode VPPT repräsentiert die Möglichkeit, verschiedene Softwarekomponenten (Texte, Bilder, Diagramme, Animationen, Video, Audio, Datenbanken, Verzeichnisse, Buttons, Befehle, Funktionen etc.) strukturiert zu einer Anwendung/Applikation zusammen zu führen, ohne dabei eine spezielle Programmiersprache einsetzen zu müssen.

Zemprosys ist als funktionierende Beta-Version verfügbar und wird derzeit noch umfangreich weiterentwickelt. Dennoch ist die Entwicklung soweit fortgeschritten, dass die Richtung der Technologie und die damit verbundene Produktionsmethode voll demonstriert und vermittelt werden können.

Dieses Dokument ist sowohl für Entwickler als auch für einfache Computernutzer vorgesehen, da wir explizit auf die Vermittlung von abstrakten Aspekten der Technologie verzichten haben. Wir hoffen jedenfalls, dass die Leser keine Schwierigkeiten haben werden, den Dokumentinhalt zu verstehen. Unsere Absichten beim Schreiben dieses Dokuments sind nicht nur die Grundlagen von Zemprosys zu vermitteln, sondern auch zu zeigen, wie und in welcher Richtung Zemprosys weiter entwickelt werden kann. Alle Alternativen zur Entwicklungsrichtung wurden marktorientiert ausarbeitet und bestimmen die zukünftigen Geschäftsfelder von **Zemprosys Group**.

Am Ende des Dokumentes befinden sich zwei Screenshots von Zemprosys. Möchten Sie mehr über Zemprosys erfahren, empfehlen wir die Webseite www.zemprosys.com zu besuchen.

# 2. MOTIVATION

## 2.1. AUSGANGSSITUATION

Als Computernutzer hat man heutzutage das Problem, dass es fast kein einheitliches und flexibles Softwaresystem existiert, mit dem man die wachsenden Aufgaben von IT-Systemen durch Selbstentwicklung, Selbstaktualisierung und Skalierung der Software gerecht werden kann. Nur Softwareentwickler können generell Computerprogramme entwickeln und durch Programmierung neuer Funktionen effizient erweitern: Eine sehr gute Erfahrung in der Programmierung ist eine relevante Voraussetzung für die Erstellung von Software.

Angenommen, man möchte ja alle seinen Kontakte über Organisationen und Unternehmen, alle seinen Aktivitäten mit Hilfe eines eigenen Computerprogramms pflegen, steuern und verwalten. Man braucht dafür eine Applikation für Adressenverwaltung, Tabellenkalkulationen, Bilderdatenbank, Video abspielen usw. Jedenfalls braucht man eine Applikation, mit der man seine Geschäfte effizient abwickeln kann. Für die Lösung dieses Problems hat man unserer Meinung nach mit dem bisherigen Stand der Technologie wirklich nur zwei effektiveren Lösungsmöglichkeiten: Sind die finanziellen Ressourcen genug vorhanden, wird ein Softwareentwickler engagiert, der das anpassende Computerprogramm entwickelt. Sind dagegen die finanziellen Ressourcen nur sehr begrenzt verfügbar, versucht man durch eine Mixlösung sein Problem in den Griff zu bekommen. Mit der Mixlösung meinen wir den Einsatz vieler Programme für verschiedene Aufgaben. Beispielweise Excel für Tabellenkalkulation, Word für Office-Dokumentverarbeitung usw. Leider ist die Mixlösung auch keine optimale Lösung, weil beispielsweise Integrationsprobleme entstehen werden und die Bedienung durch Aktivierung verschiedener Programme nicht vereinfacht wird, sondern erschwert.

Es fehlt auf dem Markt einen dritten flexiblen und starken Lösungsweg, der dem Computernutzer ermöglicht, seine eigene Applikation auch ohne Kenntnisse in der Codierung und Programmierung und ohne großen Finanzierungsaufwand selbst zu entwickeln. Kann man keinen Programmcode selbst schreiben, hat man in der Regel nur eine Alternative und zwar Erfahrene IT-Spezialisten für die Entwicklung der benötigten Applikation beauftragen. Leider ist die Entwicklung einer anpassenden Software ziemlich teuer. Sogar einige Firmen und Organisationen können sie sich maßgeschneiderte Softwarelösungen finanziell nicht leisten.

Die Software-Erstellung ist immer noch eher ein anstrengendes Handwerk im Vergleich mit den Produktionsmethoden in anderen Wirtschaftszweigen. Die textuelle Programmierung ist noch nicht Weg zu denken und beherrscht die Welt der Softwareproduktion. Überhaupt können dieselben textlastigen Methoden der Programmierung fatal auf die Komplexität eines Entwicklungsprojekts auswirken. Die Verwaltung von Softwarekomponenten und deren Abhängigkeiten führen meisten zu verketteten Änderungen der Softwaremodulen, die in bestimmten Fällen durch die textuelle Programmierung nur sehr schwer in den Griff zu bekommen sind. Die Entwicklungskosten, Entwicklungszeit und die Entwicklungsfehler sind durch die Einbindung von textbasierten Programm-Code Aspekte, die negative Folgen auf die Gesamtheit eines Softwareprojekts herbeiführen können.

## 2.2. HEUTIGE ANFORDERUNGEN AN SOFTWARE

Software übernimmt immer anspruchsvollere Aufgaben bei der Steuerung technischer Prozesse und bei der Unterstützung von Geschäftsprozessen. In den meisten Branchen ist die Wertschöpfung ohne Software nicht mehr denkbar. Die Software unterliegt daher immer mehr den Regeln anderer Produktbereiche: Hohe Qualität muss zu geringer Kosten und in kurzen Entwicklungszyklen

garantiert sein. Dies kann nur durch Wiederverwendung vorentwickelter und qualitätsgesicherter Bausteine erfolgen.

Im Bereich der Softwareentwicklung steht der Begriff „Komponente“ für Type von Bausteinen mit klar definierten Schnittstellen und Kompositionsprinzipien zur Entwicklung von Anwendungssystemen. Allerdings hat sich die Erkenntnis durchgesetzt, dass die Verfügbarkeit solcher Bausteine allein keine höhere Wiederverwendung garantiert. Vielmehr sind bestimmte architekturelle Rahmenbedingungen notwendig. Ansätze mit klar definierten Bausteintype und Kompositionsprinzipien definieren eindeutige Vorgaben für die Entwicklung und Nutzung wiederverwendbarer Bausteine.

Die Kundenbasislösungen, die meistens aus wiederverwendbaren Komponenten bestehen, sind zunehmend gefragt, weil der Entwicklungsaufwand dadurch deutlich minimiert wird. Die Kunden wollen langfristig die Kosten für IT-Lösungen minimieren, indem sie die stets wiederkehrende Programmierung der Basisfunktionalitäten als Kundenbasislösungen haben wollen. Wegen der Kosten- und Zeitfaktoren sind Softwaresysteme mit flexiblen Konfigurationsmöglichkeiten bei den Kunden beliebter geworden. Die Konfiguration der bestehenden Softwaresysteme durch integrierte Generatoren, Editoren, Designern und Wizards ist der Trend für IT-Lösungen.

Ebenfalls aufgrund der Kosteneinsparungen sind Kunden bereits, die vorhandenen Systeme selbst zu erweitern, wenn die notwendigen Änderungen und Anpassungen keine Programmierkenntnisse erfordern. Überhaupt ist die Akzeptanz für Selbsterweitbare Softwaresysteme bei den Kunden sehr groß. Standardsoftware modifizieren, erweitern und individualisieren durch selbstständige Durchführung ist ein sehr öfter geäußerter Wunsch. Flexibilität muss in Softwaresystemen vorhanden sein, damit man auf Änderungen schnell reagieren kann, ohne dabei komplexe und zeitaufwendige Entwicklungsschritte einzuleiten.
Kundenwunsch ist es, preisgünstig hochwertige Komponenten zu erwerben, die auf einer ausgereiften Technologie basieren. Und die mit geringem Aufwand an dem individuellen Bedarf angepasst werden können. Customizing ist in Zukunft wichtiger als Programmierung.

Die Sicherstellung der Portabilität und Migration der Daten von einem System zu einem anderen hat eine höchste Priorität bei den Kunden. Eine präzise, unkomplizierte und sichere Methode zum Datenaustausch ist in der Zeit der Internationalisierung von Geschäften und der wachsenden Anforderung an IT-Lösungen ein wichtiger Faktor.

Im privaten Bereich ist der große Wunsch, Computerprogramme zu haben, die die zunehmenden Aufgaben einheitlich und ohne Integrationsprobleme lösen können. Die privaten und professionellen Anwender möchten gern ihre eigenen Applikationen nach Bedürfnissen basteln, ohne dabei Programmierkenntnisse zu haben oder finanziell viel ausgeben zu müssen. Insbesondere im Engeneerings- und Industriebereich fallen die Probleme der projektbezogenen Softwarenutzung auf. Zeitkonfigurierbare IT-Lösungen für die Wiederverwendbarkeit von Informationen und für einen flexiblen Informationsaustausch zwischen Fachkräften bzw. Abteilungen werden dringend gebraucht.

Viele Problembereiche existieren, warum eine neue Produktionsweise von Software entwickeln werden muss. Jedenfalls ist die textuelle Programmierung aufgrund der steigenden Komplexität von Geschäftsprozessen heutzutage für viele Projektarten nicht mehr gerecht. Textbasierte Bausteinen (Deklarationen, Konfigurationsangaben, etc.) der Softwareproduktion sollen abgelöst werden.

# 3. SOFTWARE-ZIEL

## 3.1. PROGRAMMIERUNG DURCH DEN BENUTZER

Eigentlich weiß man schon seit Jahren, dass zwar die Implementation eines Algorithmus am besten mit sprachlichen Mitteln vorzunehmen ist, aber es ist für Entwickler schwierig, auf den Benutzer angepasste Programme zu schreiben. Könnte der Benutzer selber programmieren, so wäre er in der Lage, solche (kleine) Programme selber zu schreiben oder zumindest vorhandene Lösungen an seine Bedürfnisse anzupassen. Er müsste sich dann nicht mit Standardlösungen zu Frieden geben, sondern könnte sein individuelles Programm, angepasst an seine Bedürfnisse, selber schreiben oder zumindest bestehende Lösungen an seine Bedürfnisse anpassen. Dies hätte auch den Vorteil, dass er bei Änderungen in seinem System das Programm selbständig anpassen könnte, ohne dass die Hilfe von professionellen Entwicklern benötigt würde, womit Kosten gespart werden könnten.

Wichtig bei all der Lösungsansätzen ist es, dass der Benutzer die „Programmierumgebung“ ohne großen Lernaufwand und Vorkenntnisse effektiv nutzen kann. Ist dies nicht möglich, wird der Benutzer das System nach kurzer Zeit nicht mehr benutzen und der Nutzen bleibt aus.

(1) Eine oft verwendete Art von Programmierung durch Benutzer ist das Aufzeichnen von Makros. Der Benutzer kann das System ohne Vorkenntnisse verwenden und so wiederholende Aufgaben erledigen lassen. Das Problem mit den Makros ist die oft fehlende Verallgemeinerung, so dass nur exakte Wiederholungen gemacht werden können.

(2) Ein anderer Ansatz ist die visuelle Programmierung. Der Benutzer kann durch Zusammensetzen von gegebenen „Programmier-Bausteinen“ sein eigenes Programm schreiben. Der Benutzer muss keinen Syntax lernen. Er programmiert durch das Zusammenhängen von Bausteinen. Beispiele sind die Programmierumgebung zum Lego Mindstorm oder LabVIEW, ein Umgebung zur grafischen Erstellung von Messprogrammen.

(3) Natürliche Programmiersprachen verfolgen das Ziel, dass Benutzer die Programme in derselben Weise schreiben können, in der sie über das Problem denken. Dabei ist die ganze Programmierumgebung von großer Bedeutung.

(4) Skriptsprachen und Programmspezifische Sprachen versuchen, durch eine eingeschränkte Anzahl von Schlüsselwörtern, für den Benutzer einfach verwendbar zu sein. Die Kehrseite ist ein auf diese Schlüsselwörter eingeschränktes Anwendungsgebiet.

(5) Es gibt weitere Ansätze, wie zum Beispiel „Programming by Demonstration“. Wäre es nicht toll, wenn man dem Computer einen Arbeitsablauf einmal zeigen könnte und der Computer das ganze, bei Erkennung ähnlicher Situationen, auf die Situation angepasst wiederholen würde? Der Computer nutzt das Beispiel, das durch den Benutzer vorgegeben wurde und wendet das Gelernte auf neue Situationen an. Ziel von „Programming by Demonstration“ ist es, die Handlungen des Benutzers auf ähnliche Situationen anzuwenden. Dabei können Schwierigkeiten auftreten. Die Handlungen des Benutzers müssen korrekt interpretiert werden, d.h. das Wesentliche der Handlung muss erkannt werden. Dies ist dann die Grundlage für eine korrekte Verallgemeinerung. Oft ist es recht schwierig die wesentlichen Teile einer Handlung zu bestimmen und diese dann richtig zu verallgemeinern.

Das Ziel von Zemprosys ist es, die eben aufgelisteten Lösungsansätze - unter Berücksichtigung der heutigen Anforderungen an Software - in die Realität umzusetzen, wobei mehr Akzente auf Punkt (2) und (4) gesetzt werden.

# 4. TECHNOLOGIE

## 4.1. ÜBERBLICK

Mit Zemprosys ist es gelungen, die logische Operationen und Anweisungen, die während der Programmierung geschrieben werden durch ein visuelles, dynamisches und flexibles Produktionsverfahren zur Erzeugung von Applikationen zu ersetzen. Zemprosys ist also eine Software mit einer graphisch zu bedienenden Oberfläche für die Herstellung von Softwareanwendungen. Vom Prinzip her verfügt Zemprosys über Befehle, Funktionen und Elemente, die nach Bedarf zu einem eigenständigen Programm zusammengestellt werden können.

## 4.2. FUNKTIONSWEISE

Um eine Applikation im Kontext von Zemprosys zu entwickeln, werden 2 Software gebraucht. Eine der Software ist eine Entwicklungsumgebung (**ZDS**) und die andere repräsentiert ein Laufzeitprogramm (**ZCI**). Mit ZDS kann eine Applikation ohne Codeschreibung erzeugt werden und mit ZCI wird diese Applikation ausgeführt. Standardmäßig wird das Laufzeitprogramm kostenlos vertrieben und meistens über das Internet.

Die folgende Abbildung verdeutlicht, wie die Technologie funktioniert:

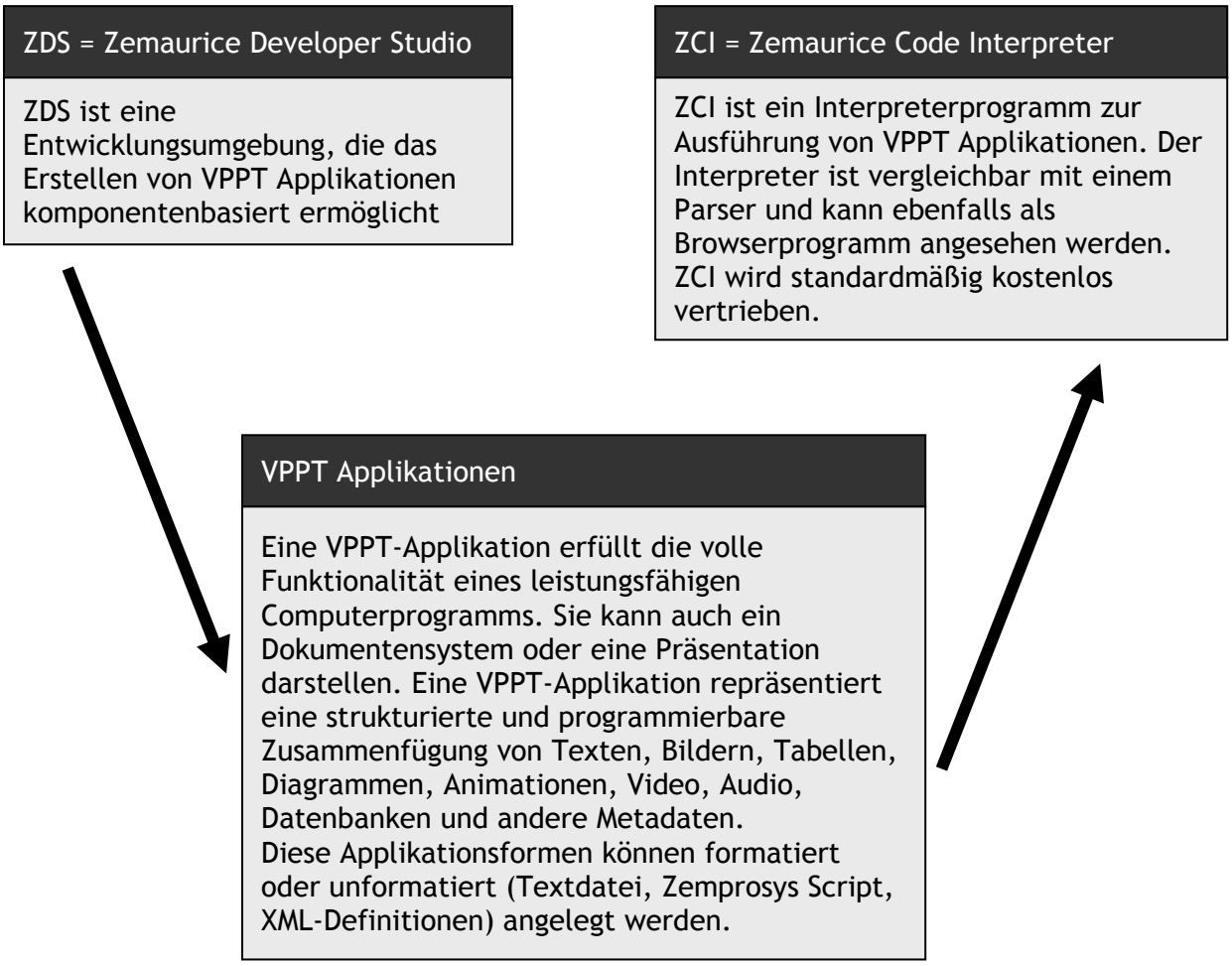



**Abbildung 1**: Funktionsweise der Technologie

Eine VPPT-Applikation ist im Grunde das Ergebnis einer Softwaremontage. Von der Softwareentwicklung wird die ganze Produktionsweise auf Montage reduziert. Und diese Montage funktioniert wie in der realen Welt: Brauche man eine Applikation, nimmt man Komponente C1, C2, C3, C4 usw. und montiere sie zusammen. Braucht man in seiner Applikation die Komponente C3 nicht mehr, kann man problemlos C3 entfernen. Möchte man seine Applikation mit C5 erweitern, integriert man C5 problemlos. Was wir hier durch Beispiele versuchen zu erklären, ist eine neue Produktionsmethode von Software.

Wenn wir die Software-Montage auf unternehmerische Bereiche übertragen, existieren eine Vielzahl von Anwendungs- Einsatzbereichen. Sofern unternehmerische Softwarekomponenten vorhanden sind, haben Firmen selbst das Instrument, um Software-Montage ohne Einsatz von hochqualifizierte Fachkräfte zu betreiben.

Es ist hinzuweisen, dass keine Compiler-Software bei dieser Softwaremontage eingesetzt wird. Weder ZDS noch ZCI bieten Compiler-Funktionen an. Zwar wird durch **ZemprosysScript** eine interne Beschreibungssprache verwendet, aber die Interpretierung dieser Sprache entspricht lange noch nicht der Logik einer Compiler-Software. Deshalb ist auch das Laufzeitprogramm ZCI zur Ausführung von VPPT-Applikationen unverzichtbar.

## 4.3. VISUAL PAGE PROGRAMMING TECHNOLOGY

Der Schwerpunkt der Technologie ist das Verfahren VPPT. VPPT ist die Abkürzung für **Visual Page Programming Technology** und bezeichnet die Möglichkeit, mittels einer graphischen Oberfläche per drag & drop Programme und aktive Dokumente zu konstruieren, ohne dabei den Programmcode selbst schreiben zu müssen. Dieser wird von der VPPT Technologie automatisch erzeugt. Das VPPT-Verfahren repräsentiert in der Entwicklungsumgebung die Möglichkeit, Texte, Bilder, Diagramme, Animationen, Video, Audio, Datenbanken, Verzeichnisse, Buttons, Befehle, Funktionen etc. strukturiert zu einer Anwendung/Applikation zusammen zu führen. Durch diesen universalen Charakter lässt sich daraus nahezu jede Art von Applikation entwickeln.

• **Portabilität**. VPPT Applikationen können flexibel eingesetzt werden, da z.B. keine komplexen Installationen notwendig sind. Sie laufen unabhängig von externen Modulen und sind verfügbar auch in Form von Texten (Textdatei). Die Portierung auf verschiedene Systeme (Win, Mac, Linux...) ist daher eine sehr einfache Sache, sofern ein entsprechender Interpreter vorhanden ist. Änderungen daran können sowohl manuell, als auch automatisch oder dynamisch vorgenommen werden.

• **Scripting**. Das VPPT-Verfahren ist mit eigenen internen Skript-Befehlen strukturiert. Die Portabilität des VPPT-Verfahrens ist teilweise gewährleistet dadurch, dass jede erstellte VPPT-Applikation vollständig in Textdatei (mit Skript-Befehlen) konvertiert und rückkonvertiert werden kann (siehe Abschnitt „Zemprosys Script“)

• **Komponentenverwendung**. Der relevante Unterschied zwischen Zemprosys und der allgemeinen Softwareentwicklung liegt darin, dass die Verwendung von Komponenten in der Softwareentwicklung meistens mit einer zusätzlichen textuellen Programmierung zur Finalisierung von Anweisungen verbunden ist, während bei Zemprosys diese Verwendung nur durch die visuelle Programmierung und ohne Schreiben eines Code stattfindet.

Zemprosys verfügt momentan über 225 Komponenten. Mindestens 85 davon können heute schon vom Anwender genutzt werden, um allgemeine eigenständige Applikationen zu erstellen. Die Komponenten werden nach dem Baukastenprinzip angewandt. Jede für sich kann jederzeit eigenständig hinzugefügt oder entfernt werden.

• **Persistent**. Die Verwendung von Komponenten ist persistent, d.h., jede verwendete Komponente kann über mehrere Anwendungssitzungen durch Speichern in Datei überdauern.

• **Objektpublizierung**. Mit der Objektpublizierung wurde ein Konzept vom Inspektorverfahren abgeleitet, das zwei wesentlichen Aspekte beinhaltet: Automatismen wurden einerseits entwickelt, mit deren Hilfe alle Komponenteneigenschaften auf die gleiche Weise bearbeitet werden. Anderseits wurde die klassische Eventbehandlung der Visualprogrammierung durch den Live-Methodenzugriff ersetzt. Die Benutzerschnittstelle für den Zugriff auf Komponenten der Software lässt sich auf Basis der Objektpublizierung realisieren.

• **Datentypskalierung**. Das VPPT-Verfahren enthält eine große Menge an Datentypen. Ein wichtiges Merkmal der Datentypskalierung hier ist ihr Editiervorgang. Alle Datentypen lassen sich sehr einfach und mit Vorschau der Ergebnisse editieren.

• **Visualdesignkonzept**. Das Visualdesignkonzept (Vcxdesign) ist ein Verfahren, das eingesetzt wird, um per Mausklick Komponenten unterschiedlich zu gestalten. Das Ergebnis dieses Verfahren ist in erster Linie eine drastische Aufwandreduzierung während der Erstellung von VPPT-Applikationen. Die Grundidee des Vcxdesign ist eine umfangreiche Einstellungsvorgabe für Komponenten verbunden mit Automatismen.

• **Containerdesign**. Ein dominierendes Element im Konzept von Zemprosys ist das Containerdesign. Mit diesem Element ist Zemprosys u.a. in der Lage, verschiedene Fenster einer Applikationen durch ein einziges Fenster zu ersetzen. Die gesamte Entwicklungsumgebung setzt auf dieses Konzept, um ohne zahlreiche Programmfenster arbeiten zu können. Mit dem Containerdesign sind ferner verschiedene Beziehungsarten von Komponenten möglich: Ausgehend von einer Mutterkomponente können u.a. Komponentenhierarchien, Iterative Zugriffe, Referenzierungen, Kollektionen, Objektbäume usw. implementiert werden.

• **Partitionsdesign**. Das Partitionsdesign wird vom Containerdesign abgeleitet mit dem Ziel, eine Applikation, ein Dokumentensystem oder eine Komponente in Partitionen zu strukturieren. Interessant dabei ist die Verknüpfung zwischen dem GUI (Graphical User Interface) und den einzelnen Partitionen, die graphisch realisiert werden kann.

• **Multifunktionalität**. Das VPPT-Verfahren ist von Multifunktionalität geprägt. Die Multifunktionalität ergibt sich dadurch, dass ein Komponententyp für verschiedene Zwecke eingesetzt werden kann. Sowohl graphisch als auch funktionell ist die Multifunktionalität Bestandteil jeder Komponentenorientiert.

• **Linkprogrammierung**. Ähnlich wie beim Internetlinkverfahren werden definierte Links eingesetzt, um einerseits Komponenten miteinander zu verknüpfen, anderseits um Dateien zu öffnen oder auszuführen. Während ein Internetlink einfach strukturiert ist, kann der Vorgang des Verlinkens in Zemprosys komplexe Strukturen aufweisen. Multiple Aktivitäten während des Linkprozesses können eingeleitet werden.
Zahlreiche Ausführungsparametern können ebenso an dem Linkprozess eingebunden werden.

• **Eventprogrammierung**. Die Eventprogrammierung repräsentiert das wichtigste Element des VPPT-Verfahrens, weil die Codeschreibung (textuelle Programmierung) von diesem Element (Eventcodierung und Eventspeicherung) ersetzt wird. Während der Verwendung von Komponenten können integrierte Komponentenaktionen programmiert werden, die dann in dem richtigen Zeitpunkt ausgeführt werden. Durch die Übergabe & Speicherung von Parameterwerten ist eine Skalierung des VPPT-Verfahrens gewährleistet.

Bei der Eventprogrammierung geht es hauptsächlich darum, wie eine Komponente C1 eine anderen Komponente C2 veranlassen kann, eine bestimmte Aktion auszuführen. Man definiert das Eventkonzept dadurch, dass C1 eine Botschaft an C2 sendet. Das Prinzip hier ist die Essenz der Objektorientierten Programmierung bzw. Visualprogrammierung. Zemprosys leitet von diesem Prinzip ein dynamisches Datencodierungsverfahren ab, um alle logischen Operationen und

Anweisungen, die für die Implementierung von Botschaftsmechanismen bzw. Eventmechanismen nötig sind, zu ersetzen. Um Events codieren zu können, müssen die Fähigkeiten des Empfängers (C2) bekannt sein. "Fähigkeiten" sind dabei die Aktionen, die vom Empfänger ausgeführt werden können.

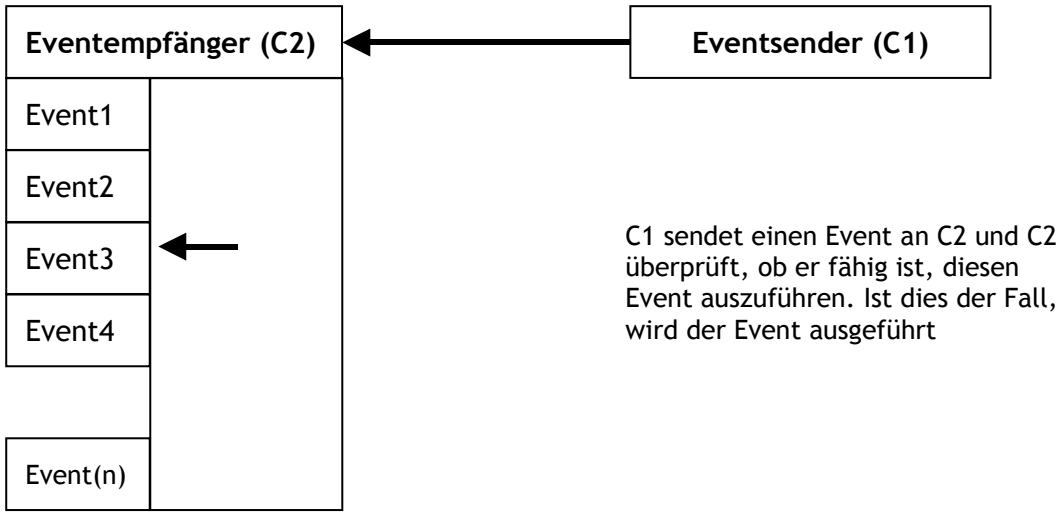



**Abbildung 2:** Prinzip der Eventbehandlung

• **Expertenprogrammierung**. Experten sind bei Zemprosys solche Komponenten, die spezielle Lösungen für bestimmte Applikationsspezifische Aufgaben anbieten. Sie besitzen die volle Funktionalität eines Computerprogramms. Der interessante Vorteil von Expertenkomponenten ist die Portabilität. Auf der Plattform Zemprosys können Expertenkomponenten uneingeschränkt wie andere Basis-Komponenten verwendet werden ( z.B. in Projektseiten anlegen, aus einer Seite kopieren, in eine Seite einfügen usw.)

• **Befehlseditor**. Der Befehlseditor ist die graphische Schnittstellen für die Bearbeitung von Befehlen des VPPT-Verfahrens. Der Befehlseditor kann auch komplexe Befehle codieren. Bei der Codierung hat der Anwender die Freiheit, einen komplexen Befehl in zahlreiche Aktionen (COM[1], COM[2]...COM[N]) zu zerlegen und die einzelnen Aktionen unterschiedlich ausführen zu lassen. Man hat somit die Möglichkeit, sequenzielle, zeitgesteuerte, synchronisierte oder blockausführbare Befehle zu generieren.

## Wo liegen die Grenzen des VPPT-Verfahrens?

Obwohl die Tragweite des VPPT-Verfahrens sehr groß ist - obwohl wir durch VPPT ein enorm wirtschaftliches Potenzial ansehen, kann die Codeschreibung während der Softwareentwicklung nur projektbezogen durch das VPPT ersetzt werden. Nicht alle Projekte lassen sich durch VPPT problemlos implementieren. Die Tatsache, dass wir hier mit einer Nichtprogrammiersprache zu tun haben, führt dazu, dass es unmöglich ist, eine Entwicklungslösung für alle Probleme zu geben. Nur durch Kombination (VPPT-Programmierung + Programmierung mit Code) können alle Type von Applikationen generiert werden. Für einige komplexe Projekte ist es daher zwingend, das VPPT-Verfahren mit zusätzlichem Programmierungsaufwand zu ergänzen.

Die Leistungsfähigkeit von Zemprosys ist abhängig von verfügbaren Softwarekomponenten. Je mehr diese Komponenten vorhanden sind, desto höher ist die Leistung der Softwaretechnologie. Um also das VPPT-Verfahren stark zu machen, müssen viele Komponenten anwendungsorientiert entwickelt werden.

Die Stärke des VPPT-Verfahrens ist ebenfalls abhängig von Komponentenaktionen. Sind diese Aktionen ausreichend vorhanden, steigt automatisch die Leistung von Zemprosys. Die Technologie ist bisher so entwickelt worden, dass diese Aktionen erst nach Marktbedarf programmiert werden.

## 4.4. ZEMPROSYS SCRIPT

Wenn mit Hilfe der Entwicklungsumgebung ZDS möglich ist, Applikationen rapide, visuell und ohne Schreibung eines Codes zu entwickeln, stellt sich dann die Frage, warum man noch eine Skriptsprache braucht, zumal auch zahlreiche Skript-Technologien existieren.

Die Gründe für eine eigene Skriptsprache sind unterschiedlich: Mit einer eigenen Skript-Technologie sollte es Wege für Open Source Projekte eröffnet werden, da in diesem Fall freiwillige Entwickler die Mittel hätten, - auch ohne die Entwicklungsumgebung - eigene VPPT-Applikationen zu erstellen. Wenn man in Betracht zieht, dass ZCI (das Laufzeitprogramm) kostenlos vertrieben wird, ist die Einführung einer eigenen Skriptsprache eine zukunftsorientierte strategische Entscheidung für eine solide Etablierung von Zemprosys auf dem Softwaremarkt. Der andere Grund, warum wir auf **ZemoprosysScript** beharren, ist durch die grundlegenden Aspekte von **Zemprosys NetEngine** gegeben, weil diese Web-Plattform mit einem Datenbanksystem verknüpft ist, dessen Tabellen die VPPT-Applikationen in Textdateiformat (Skript-Applikationen) verwalten.

### (1) Das Wesentliche von ZemprosysScript

ZemprosysScript ist eine Sprache, deren Grundstruktur und Syntax sehr einfach sind. Die Sprache ist komponentenbasiert und kennt bis dato meistens nur die Zuweisung als Anweisung, wobei Komponenteneigenschaften Werte zugewiesen werden. Die Werte, die zugewiesen werden können, sind Zahlen, String, Farbenwerte, Komponentenzustände usw.

Wenn man sich nur auf diese einfache Anweisung (Zuweisung) beschränkt, sollte man eigentlich nicht von einer Skriptsprache reden, da beispielsweise bedingte Anweisungen, Switch-Anweisungen, while-Schleife, do-Schleife oder for-Schleife direkt nicht formuliert werden können. Ebenfalls sind direkte Funktions- und Prozeduraufrufe nicht möglich (Funktionen und Prozeduren können nur indirekt durch eine Zuweisung aktiviert werden).

Dennoch haben wir hier mit Programmierung zu tun: Aufgrund der komponentenbasierten Sprache richtet sich der Entwickler an den Anwendungsbereichen von Komponenten, um Skript-Applikationen bzw. VPPT-Applikationen zu erstellen. Jede Softwarekomponente verfügt über ausreichende Funktionen, die vom Entwickler genutzt werden können.

Was vom Entwickler als primäre Voraussetzung verlangt wird, ist die Komponentenbibliothek von Zemprosys umfangreich zu kennen, indem der Entwickler wissen sollte, wie Eigenschaften und Methoden auf Komponenten wirken.

In der Regel schreibt man unter Anleitung von **ZemprosysDecriptor** (siehe nächsten Abschnitt) die Skript-Applikationen.

### (2) Struktureller Aufbau einer Skript-Applikation

Auch wenn wir eben nur von einfacher Zuweisung als einzige wirkliche Anweisung geredet haben, verfügt eine VPPT-Applikation (in Skript-Form) über weitere Anweisungen (**generische Anweisungen**), die dazu dienen, die Applikation während des Ladenvorgangs zunächst ohne Applikationslogik zu rekonstruieren. Unter anderem werden Komponentenreferenzen auf Basis generischer Anweisungen gesetzt.

Die folgende Zeilen spezifizieren den allgemeinen strukturellen Aufbau einer VPPT-Applikation in Skript-Form:

```
BeginInterface
        Zuweisung...
        Zuweisung...
        ...
EndOfInterface

BeginDeclaration
        Class = ObjectName: MyIden
        EndOfClass

        Class = ObjectName: MyIden
                Child = A
                Child = B
                Child = C
        EndOfClass

        Class = ObjectName: MyIden
        EndOfClass
EndDeclaration

BeginObject
        <MyIden> = A
        <ObjectName> = B
        <AlliasName> = C
        Zuweisung...
        Zuweisung...
        Zuweisung...

        BeginStructure: Name
        EndOfStructure

        ColorAlpha: Name
        EndOfList

        StructureAsFont: Name
        EndOfStructure

        StructureAsBrush: Name
        EndOfStructure

        StructureAsBorder: Name
        EndOfStructure

        StructureAsColor: Name
        EndOfStructure

        StructureAsImage: Name
        EndOfStructure

        StructureAsRelation: Name
        EndOfStructure

        BeginSimpleList: Name
        EndOfList

        BeginComplexList: Name
        EndOfList

        BeginCollection: Name
```

```
            EndOfCollection

        EndObject

        BeginObject
        EndObject

        BeginObject
        EndObject
        ...
```

Beim vorliegenden Dokument präsentieren wir nur ganz kurz den strukturellen Aufbau einer Skript-Applikation. Eine ausführliche Beschreibung wird zusammen mit ZemprosysDescriptor auf einem anderen Dokument vorgenommen.

**BeginInterface ... EndOfInterface**
Definiert einen Block von Zuweisungen, die zur allgemeinen Initialisierung der Skript-Applikation dienen. Beispielweise kann die Standardüberschrift (Titel) oder die Hintergrundfarbe der Applikationen in innerhalb dieses Blocks zugewiesen werden.

**BeginDeclaration ... EndDeclaration**
Dieser Block ist für die Durchführung von Deklarationen vorgesehen. Alle Applikation-Komponenten werden hier registriert und zwar mit Hilfe einer Liste von Class...EndOfClass-Anweisungen. Sollen beispielsweise 5 Komponenten verwendet werden, müssen daher auch 5 Class...EndOfClass-Anweisungen geschrieben werden.

**Class ... EndOfClass**
Eine Komponente wird für eine Applikation registriert, indem die Anweisung Class...EndOfClass geschrieben wird. Wie der strukturelle Aufbau es zeigt, besitzt diese Anweisung zwei wichtigen Parametern: **ObjectName** gibt den Name der Komponente an und **MyIden** die Identifikation der Komponente. Der Komponentenname ist freiwillig zu bestimmen, wie z.B. „MeineListe“ und muss innerhalb der Applikation **eindeutig** sein. Die Identifikation MyIden ist eine Zahl, die auf eine kompilierte bzw. binäre Komponente der Komponentenbibliothek verweist. Mit Hilfe von ZemprosysDescriptor kann diese Zahl unkompliziert und anschaulich ermittelt werden. Daher kann MyIden nicht einfach so festgelegt werden (diese Zahl muss in der Zemprosys-Architektur existieren und wird vom Komponentenentwickler festgelegt)

Sofern eine Komponente in der Lage ist, andere Komponenten aufzunehmen - es handelt sich hier um Container - und aufnehmen sollte, wird das Schlüsselwort **Child** innerhalb der Class...EndOfClass-Anweisung verwendet. Durch Schreiben von Child = A, wobei A den Name einer deklarierten Komponente repräsentiert, spezifiziert man, dass die Komponente A als Kind eine andere Komponente registriert werden muss. Man muss allerdings darauf achten, dass die gleiche Zuweisung Child = A nicht an mehreren Stellen vorkommt und zugleich dass die Komponente A tatsächlich auch deklariert ist.

Mit dem Schlüsselwort **Child** können Referenzen auf Referenzen gebildet werden und somit verschachtelte Container-Systeme generiert werden.

**BeginObject ... EndObject**
Jede deklarierte Komponente muss mit der BeginObject...EndObject-Anweisung strukturell und funktionell aufgebaut werden. Innerhalb des Blocks können einfache Zuweisungen, Struktur- und Kollektion-Anweisungen vorgenommen werden. Für jeden Aufbau einer deklarierten Komponente sind die Zuweisungen **<MyIden> = A** und **<ObjectName> = B** pflicht. A und B entsprechen den Werten, die als Parameter für Class...EndOfClass festgelegt sind.

**BeginCollection ... EndOfCollection**

Die Anweisung BeginCollection...EndCollection lässt das Schreiben einer Liste von Strukturen zu. Mit anderen Worten, innerhalb dieses Blocks dürfen nur Anweisungen der Form BeginStructure...EndOfStructure getragen werden. Sowohl homogene als auch heterogene Kollektionen können aufgebaut werden. Auch hier bestimmt der Komponententyp die Art der Kollektion, die aufgebaut werden kann.

**BeginStructure ... EndOfStructure**
Dient zur Beschreibung einer generischen Struktur bzw. eines Datensatzes. Innerhalb des Blocks dürfen nur einfache Zuweisungen geschrieben werden.

**BeginSimpleList ... EndOfList**
Die Anweisung BeginSimpleList ... EndOfList wird benutzt, um eine Liste von Werten in der Form **[] = Value[i]** zu schreiben, wobei i den Index eines Listenwertes repräsentiert.

**BeginComplexList ... EndOfList**
Ermöglicht ebenfalls das Schreiben einer Liste, wobei die einzelnen Zuweisungen innerhalb des Blocks die Form **Bez[i] = Value[i]** besitzen: Bez[i] gibt die Bezeichnung eines Listenelementes an und Value[i] dessen Wert.

**ColorAlpha ... EndOfList**
Diese Anweisung wird benutzt, um eine Farbenliste zu schreiben. Jeder Farbeneintrag wird durch eine Buchstabe identifiziert. Es können höchsten 26 Farbeneinträge spezifiziert werden und die zulässige Form der Zuweisung ist **[Buchstabe] = Value**, wie z.B. [A] = Color1 oder [B] = Color2

**StructureAsFont ... EndOfStructure**
Wird verwendet, um Schriftsinformationen (Schriftart, Schriftfarbe, Schriftgröße etc. ) zu schreiben

**StructureAsBrush ... EndOfStructure**
Wird verwendet, um eine Schattierungsmuster zu schreiben.

**StructureAsBorder ... EndOfStructure**
Wird verwendet, um eine Rahmenart zu spezifizieren.

**StructureAsColor ... EndOfStructure**
Wird benutzt, um einen Datensatz bestehend aus Farben zu spezifizieren.

**StructureAsImage ... EndOfStructure**
Die Anweisung StructureAsImage ... EndOfStructure lässt das Schreiben von Bilderinformationen als Datensatz zu.

**StructureAsRelation ... EndOfStructure**
Diese Anweisung wird eingesetzt, um eine graphische Relation zu spezifizieren.

## (3) Hinweise

- Eine VPPT-Applikation kann als eine Tabelle mit einer Spalte verstanden werden, wobei die Struktur jeder Tabellenzeile durch eine deklarierte Komponente bestimmt wird. Die Reihenfolge der Class...EndOfClass-Anweisungen ist daher sehr relevant.

- Für den strukturellen und funktionellen Aufbau einer Skript-Applikationen mit Hilfe der Anweisung BeginObject ... EndOfObject verfügt jeder Komponententyp über eine **Skript-Vorgabe**. Die Aufgabe eines VPPT-Applikation-Entwicklers besteht in diesem Zusammenhang meistens darin, die Skript-Vorgabe nach Anforderungen anzupassen. Der Inhalt jeder Skript-Vorgabe ist aus der Beschreibung von ZemprosysDecriptor zu entnehmen.

- Über die Entwicklungsumgebung ZDS können VPPT-Applikationen erzeugt werden und in Skript-Form abgelegt werden. Skript-Applikationen wiederum können geladen und visuell programmiert werden.

- Die Entwicklung der Skript-Technologie steht noch in ihrer Frühphase. Sicherlich wird der strukturelle Aufbau einer Skript-Applikation in Zukunft mit weiteren **generischen Anweisungen** unterstützt.

## 4.5. ZEMPROSYS DESCRIPTOR

ZemprosysDescriptor ist ein Begriff zur Beschreibung der Architektur von Zemprosys. Es handelt sich dabei um ein freies Framework. Ohne dieses Framework ist die Erstellung von VPPT-Applikationen sowohl visuell als auch durch die Skript-Befehle fast nicht realisierbar.
ZemprosysDescriptor definiert ein Dokumentations- und Hilfesystem zur Unterstützung von Entwicklern durch exakte Beschreibung von Zemprosys-Elementen (Komponenten, Funktionen, Methoden, Events, Eigenschaften, Datentype, visuelle Programmierungsschnittstellen usw.)

## 4.6. ZEMPROSYS NET ENGINE

Zemprosys NetEngine ist ein System, das aus zwei wichtigen Komponenten zusammengesetzt ist (Siehe untere Abbildung). Wir reden in diesem Fall von einem **Frontend-Backend-System** basierend auf VPPT, wobei MySQL das Backend darstellt und Zemprosys das Frontend. Das primäre Ziel, das durch den Aufbau eines solchen Systems erreicht werden soll, ist die Realisierung einer Netzwerkplattform (Intranet oder Web) für die Anwendung des VPPT-Verfahrens:

- Durch die Integration von MySQL als Backend-System zielen wir auf eine netzwerkorientierte Verknüpfung mit der momentan populärsten Open-Source-Datenbank der Welt. Wir nutzen einen der wichtigsten Aspekte von Zemprosys, um dieses Vorhaben umzusetzen. Durch totale Konvertierung einer VPPT-Applikation in Textformat (Skript-Form) und Rückkonvertierung durch Ausführung der Applikation ist es möglich, ein Zemprosys-Projekt vollständig in einer Datenbank zu speichern - in diesem Fall in MySQL-Datenbank.

- Standardmäßig bietet MySQL zahlreiche Anwendungsmöglichkeiten, so z.B. eine sichere und vereinfachte Netzwerkfunktionalität. Somit kann Zemprosys als Frontend-System eingesetzt werden, um die in einer MySQL-Datenbank gespeicherten VPPT-Applikationen (Skript-Applikationen) mit Hilfe von SQL-Anweisungen aufzurufen und auszuführen.

Was man als Software für Zemprosys NetEngine benötigt, ist einerseits den MySQL Server, der bekanntlich kostenlos vertrieben wird. Anderseits wird für die Client-Software (Frontend-System) ZCI oder ZDS vorausgesetzt. Während mit ZCI der Benutzer VPPT-Applikationen weder verändert noch neu entwickelt kann, erlaubt ZDS die Erstellung von VPPT-Applikationen auf Client-Ebene und deren Speicherung auf Server-Ebene. Generell ist ZCI kostenlos zu haben.

Zemprosys NetEngine liefert die Technologie, um die Daten und Software eines Kunden bzw. eines Unternehmen zentral zu verwalten. Software sämtlicher Arbeitsplatzrechner werden zentral gemanagt (Auf einem Server). Dies ermöglicht die genaue Zuteilung von Applikationen an Mitarbeitern (Clients). Praktisch für den einzelnen Mitarbeiter, egal, an welchem Arbeitsplatz des Büros er sich einloggt: Er findet die von ihm angewendete Applikation sofort wieder in dem Status vor, in dem er sie zuletzt genutzt hatte, einschließlich persönlicher Arbeitsoberfläche. Anwendungen auf dem zentralen Application Server können leicht gepflegt werden. Um Installation, Updates, Viren oder Spyware muss sich der Nutzer keine Sorgen machen.

## Welche Funktionen sind mit Zemprosys NetEngine zu erwarten?

- Entwicklung von Datenbanken und Webanwendungen
- Implementierung von Workflows
- Einfache Ausführung von Software
- Einfache Netzwerkadministration
- Strikte Trennung zwischen Daten und Applikationen
- Mehr Transparenz, mehr Sicherheit und mehr Zuverlässigkeit
- Transaktionsmanagement und Verteilung
- Plattformunabhängigkeit
- Unkomplizierte Migration von Daten und Integration mit vorhandenen Datenbeständen
- Skalierbarkeit von Applikationen
- Personalisierung und Berechtigungsverfahren
- Versionierung von Projekten und Dokumenten
- Content Management System (CMS)
- Document Management System (DMS)
- Realisierung von Web-Kollaborationen
- Erstellung von Prototyping
- Erstellung von Business Objekten

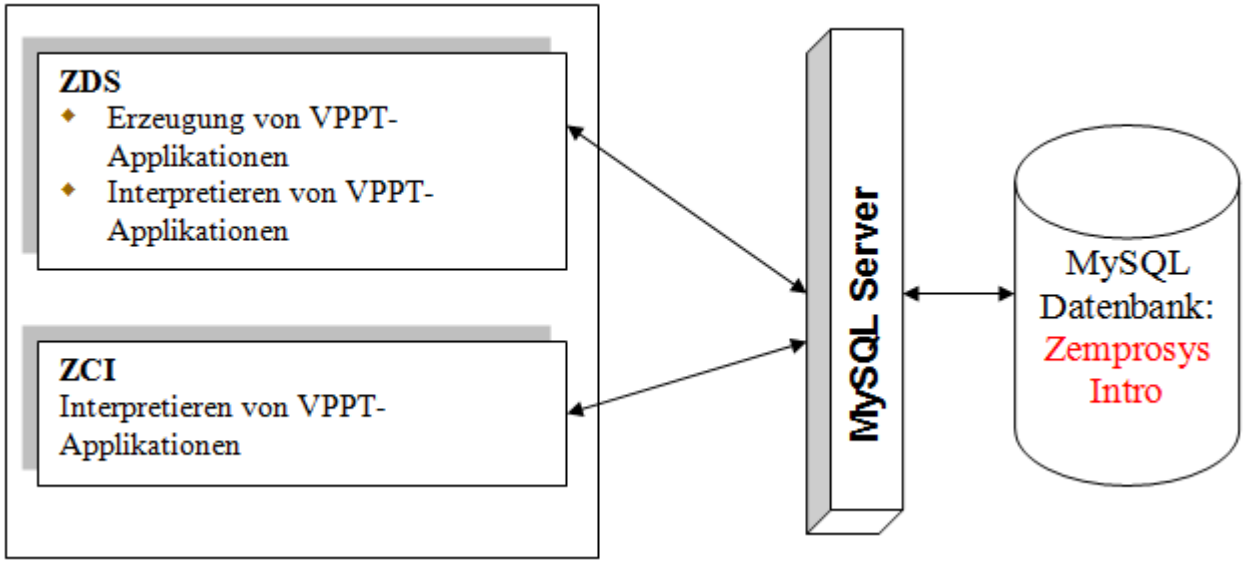



**Abbildung 3:** Frontend-Backend-System /Zemprosys NetEngine

## 4.7. PLATTFORMEN

Mit dem aktuellen Entwicklungsstand der Technologie, können Zemprosys und ihre sekundären Produkte nur unter dem Windows-Betriebssystem (Windows Vista oder höher) laufen. Wir sind also noch nicht weit mit der Entwicklung, um eine Plattformunabhängigkeit zu gewährleisten. Der Plan sieht vor, dass Zemprosys mindestens auch unter Linux-Betriebssystem laufen muss. Am bestens wäre es, wenn die Technologie für Windows, Linux und Mac verfügbar gemacht wird.

Für Entwickler von VPPT-Applikationen in Skript-Form stellt sich das Probleme der Plattformabhängigkeit für eine Skript-Applikation nicht. Die gleiche Skript-Applikation, die auf Windows läuft, wird auch auf Linux und Mac laufen. Man braucht dafür nur eine Linux- bzw. Mac-Version des Laufzeitprogramms ZCI.

Die Lösung der Plattformabhängigkeit beschreiben wir später in diesem Dokument im Kap. „Forschung & Entwicklung“.

### 4.8. INTEGRATION IN DELPHI

Da Zemprosys mit Delphi Compiler (Object Pascal) entwickelt wird, lässt sich die gesamte Technologie problemlos in Delphi integrieren. Das Herz der Technologie kann in Delphi-Plattform als visuelle Delphi-Komponente registriert werden.

## 5. ANWENDUNGEN

### 5.1. ÜBERBLICK

Das Ziel von Zemprosys ist die Entwicklung von Computerprogrammen auf Basis eines neuen Verfahrens (VPPT). Dieses Verfahren funktioniert mit „abstrakten Komponenten“, d.h., dieses Verfahren kennt nur einen Komponententyp als Basiskomponente und es ist also möglich, auf jeden neuen Komponententyp das VPPT-Verfahren anzuwenden. Ganz einfach gesagt, soll Zemprosys Datenbankprogramme unterstützen, müssen geeignete Komponenten für Datenbanksysteme entwickeln werden; soll Zemprosys GPS-Anwendungen unterstützen, müssen geeignete GPS-Komponenten in die Softwarearchitektur von Zemprosys integriert werden. Für alle Arte von Komponentenunterstützungen bleibt das VPPT-Verfahren unverändert und funktioniert auf jeden Komponententyp fehlerfrei. Somit schließen wir, dass alle erdenkbaren Anwendungsbereiche im Grunde mit Zemprosys erstellt werden können. Man muss nur die geeigneten Softwarekomponenten dafür entwickeln.

Die Anwendungsmöglichkeiten des Produktes sind so vielfältig, dass sie in ihrer Gesamtheit nur zu einer übergeordneten Aussage zusammengefasst werden können. Daher werden hier nur einige der möglichen Anwendungsbereiche aufgelistet. Insgesamt geht es immer darum, mit Zemprosys die benötigten Programme oder IT-Systeme von Grund auf zu erstellen/schreiben und zu implementieren:

- Betriebs- und Softwaresteuerungen für Maschinen, Roboter, Embedded Systeme, Terminals...
- Programme für Endanwender und IT-Strukturen nach genauen Kundenwünschen programmieren, z.B. CRM Systeme, Buchhaltungsprogramme, ERP-Software oder auch Web-Anwendungen etc.
- Eigene Programme wie z.B. ein Haushaltsbuch für die privaten Finanzen, Bilderverwaltung...

### 5.2. VISUALISIERUNGSSYSTEME

Durch die komponentenbasierte Architektur von Zemprosys und durch die zahlreichen visuellen Komponenten können wiederum Visualisierungssysteme schnell und unkompliziert erzeugt werden. Für industrielle Anwendungen und Embedded Systeme repräsentiert das VPPT-Verfahren ein besonders praktisches Instrument.

#### Beispiel: Softwareprojektierung in der Industrie

*(Industrial IT, Industrie PCs, Mobile Boxen, Gerätesoftware, Automaten, Zugriffskontrolle, Zeiterfassung, Maschinen- & Anlagenbau, Visualisierung von Prozessen)*

Die Softwareprojektierung ist ein Mittel, um interdiziplinäre Arbeitsbereiche automatisch zu binden ohne große aufwand an Codierung:

a) Die Komponentenprojektierung zur Durchführung von Signalverarbeitungen
b) Die Parameterprojektierung zur Tragung von Parametern der Ablaufsteuerung
c) Konvertierung von Windows- programmierten Elementen bzw. Komponenten
d) Programmierung anpassender Bildschirmmasken
e) Programmierung von GUI-Komponenten (Button, Screen, Container, Explorer, Forms usw.)
f) Möglichkeit des Hinzufügen und Entfernen von Eingabe und Anzeigefeldern ohne Sourcecode zu verändern
g) Mögliche Standardisierung durch eine proprietäre Programmiersprache (Scripting/ZemprosysScript)
h) Individuelle Programmanpassungen je Kunde (Scripting, Befehle, Events, Erstellen von Listen & Auswertungen ,Bildschirmmasken, Datenbanklayouts, Formulardefinitionen)

## 5.3. DYNAMISCHE DOKUMENTE

VPPT ermöglicht die Einführung und Umsetzung einer neuen Technik zur Erstellung von dynamischen Dokumenten. Das Ziel der Technologie in diesem Zusammenhang liegt darin, dass der Anwender selbst entscheidet, wie seine Dokumente sich verhalten sollen. Der Anwender bestimmt den Umfang und die Aufgaben der Funktionen, die er dem Dokument zuweist. Ein dynamisches Dokument erfüllt die volle Funktionalität eines Computerprogramms und bietet daher mehr Möglichkeiten als ein herkömmliches Dokument.

## 5.4. XML APPLIKATIONEN

(XML: Extensible Markup Language)

Im Rahmen der Erstellung von dynamischen Dokumenten trägt Zemprosys dazu bei, offene Schnittstellen (XML) zu nutzen. Durch den integrierten **XML-Generator** und **XML-Interpreter** stellt Zemprosys einen effizienten und einfachen Weg zur Verfügung, um auch komplexe offene Schnittstellen zu implementieren. XML-basierten Anwendungen mit grafischer Benutzerschnittstelle (GUI) können mit Zemprosys realisiert werden: Sowohl dokument- als auch datenbankbasierte Anwendungen sind möglich.

## 5.5. WORKFLOW APPLIKATIONEN

Zemprosys eignet sich auch hervorragend zum Entwickeln von komplexen Geschäftsprozessen (Workflows), wobei die Integration von grafischen Formularen auf Benutzerseite durch das VPPT-Verfahren problemlos möglich ist.

## 5.6. UML/AML

(UML: Unified Modeling Language - AML: Agent Modeling Language)

Die UML/AML dient heutzutage der Beschreibung von Datenbankanwendungen, Echtzeitsystemen, Grafikprogrammen, Workflow-Anwendungen und vielen weiteren Anwendungen. Kurz gesagt: Alle Softwaresysteme sollen mit der UML darstellbar sein. Mit UML/AML ist es möglich, verschiedene Entwicklungsstufen eines Softwareprojektes in Diagrammen darzustellen.

UML/AML als Modellierungssprache lässt sich in Zemprosys-Technologie integrieren und zwar durch die komponentenbasierte visuell Programmierung und das VPPT-Verfahren.

## 5.7. CBT/WBT

(CBT: Computer Based Training - WBT: Web Based Training)

Ein weiteres Anwendungsgebiet von Zemprosys ist der Aufbau von Trainingsplattformen, insbesondere bei der Entwicklung eines **Learning Management Systems** (Erstellung von digitalen Inhalten, Präsentation von Inhalten, Datenbasis aller Unterrichtsunterlagen, Unterrichtsdokumentationen, Aufgaben- und Übungswerkzeuge, Kommunikations- und Kollaborationswerkzeug, Administration des Unterrichts, Evaluation/Bewertungshilfen usw.)

## 5.8. META KOMPONENTEN

Unter Metakomponenten („Komponente über Komponente“) wird eine unternehmensorientierte Entwicklung von Komponenten verstanden, um das Softwaresystem eines Unternehmens komponentenweise zu modellieren und zu implementieren.
Die folgende Abbildung zeigt eine Strukturierungsweise eines Unternehmens:

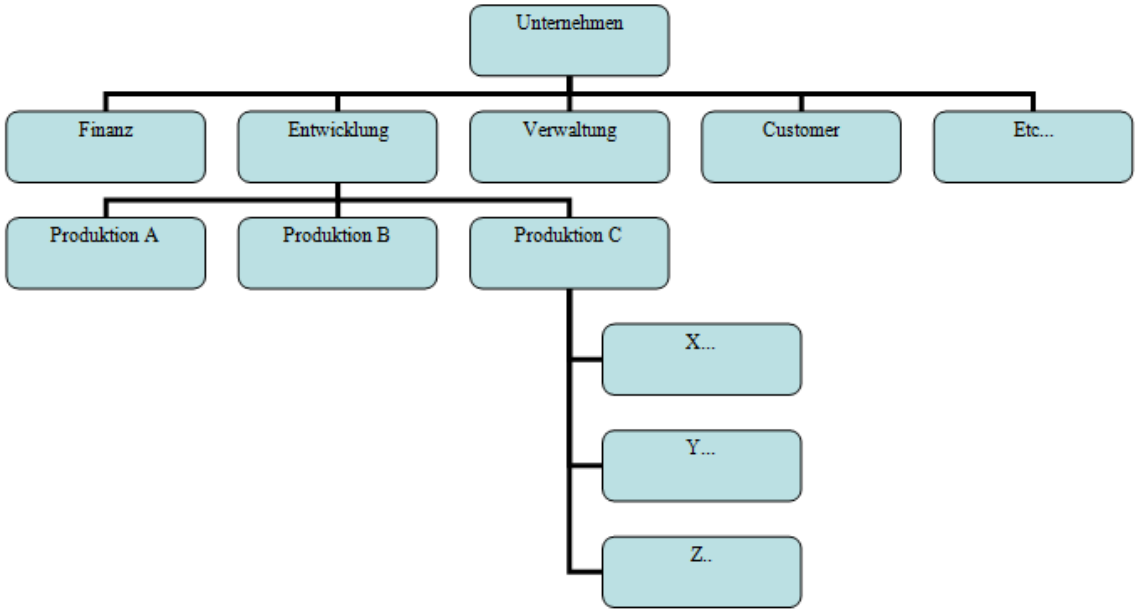



**Abbildung 4:** Strukturierungsweise eines Unternehmens

Wenn man diese Strukturierung auf das Softwaresystem eines Unternehmens überträgt, könnte man daraus schließen, dass dieses Unternehmen Software für Finanz, Entwicklung, Verwaltung, Customer usw. benötigt. Aus der Sicht von Zemprosys sind die einzelnen Strukturbereiche des Unternehmens Meta-Strukturen, die von Metakomponenten softwaremäßig gesteuert werden.

Eine Metakomponente im Sinne von Zemprosys ist eine unternehmensorientierte Zusammensetzung von mehreren einzelnen Komponenten, um eine bestimmte Unternehmensaufgabe zu erfüllen. Ein Kundenadressenformular repräsentiert beispielsweise eine Metakomponente zum Registrieren von Kunden-Kontaktdaten in die Datenbank.
Metakomponenten können auf zwei Wege erzeugt werden:

a) **Binäre/Kompiliert**: Die Metakomponente wird vollständig mit Hilfe vorhandener einzelner Komponenten programmiert (Code-Schreibung), kompiliert und in Zemprosys integriert. Auf dieser Weise ist die Metakomponente nur in binärer bzw. kompilierter Form erhältlich.

b) **VPPT-Verfahren**: Ohne Einsatz eines Codes wird die Metakomponente visuell programmiert durch Einsatz der Entwicklungsumgebung ZDS. Die Metakomponente ist auf dieser Weise in Datei-Form (formatiert oder Skript-Datei) erhältlich.

Die binäre bzw. kompilierte Metakomponente hat den Vorteil, dass durch die klassische Programmierung die Schwäche des VPPT-Verfahrens aufgehoben werden. Man kann genau programmieren, was man möchte. Auf Kosten-Ebene sollte eine nicht kompilierte Metakomponente kostengünstig sein.

Da eine kompilierte Metakomponente automatisch über ZDS verfügbar ist und aus der Sicht des VPPT-Verfahrens als einfache Komponente behandelt werden kann, können Metakomponenten einen bedeutenden Stellenwert für die Geschäftslogik eines Unternehmens einnehmen. Durch das einfache Hinzufügen und Entfernen von Metakomponenten (Baukastenprinzips von Zemprosys) während der Zusammensetzung einer Zemprosys-Applikation können Unternehmen ihre Softwaressysteme unkompliziert zusammenbauen und jederzeit aktualisieren.

# 6. FORSCHUNG & ENTWICKLUNG

## 6.1. ÜBERBLICK

Ausgehend von der Basisarbeit, die schon durchgeführt wurde und die sich durch die innovative Produktionsmethode VPPT (Visual Page Programming Technologie) zusammenfassen lässt, soll die Technologie insgesamt auf einem deutlich nutzbaren Niveau für Anwender positioniert werden. Unter anderem ist die Durchführung der zweiten Entwicklungsphase für Anwendungsorientierte Softwarekomponenten notwendig. Die Entwicklungs- und Forschungsarbeiten sollen dafür sorgen, dass das innovative Verfahren VPPT auf die verschiedenen IT-Systeme angewendet wird.

Auch wenn das Vorhaben „Programmierung ohne Code“ für bestimmte Projekttype nicht 100% umgesetzt werden kann, bietet das VPPT-Verfahren aufgrund der hohen Flexibilität, die damit verbunden ist, zahlreiche Anwendungsmöglichkeiten. Deshalb zielen die Entwicklungs- und Forschungsarbeiten auf die Integration von Standardkonzepten (DBMS, WFMS, XML, SOA, UML, CMS, DMS, CBT, WBT usw.) in Zemprosys. Diese Aufgabe ist überwiegend durch komponentenbasierte Konzepte und automatisierte Abläufe der Zemprosys-Entwicklung durchzuführen.

Kurz gesagt: Das Ziel der Entwicklung & Forschung ist ein anwendungsorientierter Einsatz des VPPT-Verfahrens durch selektive Anbindung der Technologie an die Unternehmensprozesse.

## 6.2. ENTWICKLUNGSSTAND

Zemprosys befindet sich in der Beta-Version und verfügt momentan über 225 Komponenten. Mindestens 85 davon können heute schon vom Anwender genutzt werden, um allgemeine eigenständige Applikationen zu erstellen. Die Komponenten werden nach dem Baukastenprinzip angewandt. Jede für sich kann jederzeit eigenständig hinzugefügt oder entfernt werden.

Alle Basiskonzepte zur rapiden Erweiterung der Technologie sind fast komplett implementiert worden. Insbesondere das Konzept zum Einbau neuer Komponenten ist mit einem flexiblen Verfahren entwickelt worden, sodass es relativ leicht ist, die Technologie mit neuen Komponenten zu erweitern. Einige Entwicklungsabläufe sind automatisiert und reduzieren somit den Entwicklungsaufwand von neuen Komponenten. Ebenfalls sind Automatismen gegeben, die eine klare Trennung zwischen der Weiterentwicklung von ZDS + ZCI und der Entwicklung neuer Komponenten möglich machen. Der Komponentenentwickler profitiert insofern davon, weil sich

seine Aufgaben nur auf die Anforderungen und Funktionalität von Komponenten beschränken. Der braucht sich nicht mit den Entwicklungsaufgaben von ZDS oder ZCI beschäftigen.

Das VPPT-Verfahren beinhaltet eine dynamische und marktorientierte Vorgehensweise zur Implementierung neuer VPPT-Funktionen. Insbesondere die Events (Komponentenaktionen), die das VPPT-Verfahren prägen, sind mit einem dynamischen Erweiterungskonzept verknüpft.

Die Entwicklung der Skript-Technologie ist sehr weit fortgeschritten, da die Erstellung von Skript-Applikationen schon jetzt möglich ist. Wie wir es schon in diesem Dokument im Abschnitt „Zemprosys Script“ hingewiesen haben, lassen sich die Skript-Applikationen auch visuelle erstellen, indem die VPPT-Applikationen einfach mit Hilfe der Entwicklungsumgebung ZDS erzeugt werden und dann in ZemprosysScript konvertiert werden.

Die Entwicklungsumgebung (ZDS) trägt zur kontinuierlichen und sicheren Entwicklung der Technologie bei. ZDS dient zuerst zum Testen von neuen programmierten Funktionen und Komponenten. Dadurch stellt man sicher, dass das gesamte System (Modularisierung) funktioniert. Fehler können zwar total nicht ausgeschlossen werden, aber durch „programmieren, visualisieren und testen“ reduziert sich deutlich die Anzahl der Programmfehler. ZDS stellt somit ein Repertoire zur Gewinnung von Erkenntnissen dar.

Der momentane Codeumfang von Zemprosys beträgt ca. 450 000 Programmzeilen.

## 6.3. SOFTWAREARCHITEKTUR

In diesem Abschnitt möchten wir einige wichtigen Aspekte der Architektur von Zemprosys preiswert geben. Insbesondere den Stellenwert von Komponenten soll hier grob quantifiziert werden.

Zemprosys besitzt eine komponentenbasierte Architektur. Zugriffe auf Funktionen der Technologie sind nur dank der Komponentenmodellierung möglich. Dank der objektorientierten Programmierung und der damit verbundenen Komponentenhierarchie stellt die Technologie eine konkrete Vorgehensweise bereit, um die Software rapide weiter zu entwickeln. Es existiert somit eine gut funktionierende Methodik und ein vorgegebenes Softwaredesign mit Regeln zum Updaten von Zemprosys. Der Entwicklungsprozess kann dadurch nur positiv beeinflusst werden.

Alle methodischen Aspekte zur Weiterentwicklung von Zemprosys, sind im Erweiterungskonzept von Zemprosys integriert. Dieses Konzept ist von folgenden Elementen dominiert:

- **Vererbungsmechanismus**. Das Fundament der Technologie ist eine eigene Komponentenfamilie. Das interne Leben von Zemprosys ist von Vererbungsmechanismen geprägt. Am Anfang der Komponentenhierarchie steht eine Basiskomponente und alle anderen Komponenten werden von dieser Basiskomponente abgeleitet. Die Basiskomponente wurde unabhängig von existierenden Softwarekomponenten dritter Hersteller entwickelt. Die Mächtigkeit dieser Komponente und ihre Unabhängigkeit machen Zemprosys als eine sehr flexible Technologie. Obwohl Zemprosys auf die Plattform des **Borland Delphi Compilers** entwickelt wird, sind Komponenten von Zemprosys nicht abhängig von Delphi-Komponenten und verfügen über ihre eigenen Fähigkeiten.

- **Virtuelle Komponententabelle**. Jede Komponente besitzt eine **eindeutige** Identifikationsnummer (ID = MyIden). Die Komponenten-IDs werden über eine virtuelle Tabelle verwaltet. Alle Erzeugungsaktivitäten für Komponenten sind Dank der virtuellen Tabelle vereinfacht. Auch die Objektpublizierung - eigentlich Komponentenpublizierung - profitiert enorm von der virtuellen Tabelle. Die Komponentenpublizierung macht es möglich, dass ein interaktiver Prozesse zwischen Benutzer und Komponenten aufgebaut wird.

- **Komponentenfähigkeiten**. Die Fähigkeiten einer Zemprosys-Komponente sind fast unbegrenzt. Man kann sogar eine Zemprosys-Komponente als ein eigenständiges Computerprogramm modellieren. Hier sind einige Fähigkeiten:

  a) Eine Komponente kann sich selbst zeichnen und somit auf dem Bildschirm dargestellt werden
  b) Eine Komponente kann ihre Daten in einer Datei schreiben
  c) Eine Komponente kann ihre Daten aus einer Datei laden
  d) Eine Komponente kann ihre Daten drucken
  e) Eine Komponente erbt automatisch von Zemprosys-Automatismen, damit sie über ZDS editiert werden kann (Einstellung von Eigenschaften, Methoden-Aufrufe, VPPT-Programmierung usw.)
  f) Eine Komponente erbt automatisch von Zemprosys-Automatismen, um Skript-Vorgabe zu generieren. Damit wird es sichergestellt, dass eine Komponente in Skript-Form geschrieben werden kann und von einer Skript-Datei interpretiert werden kann
  g) Eine Komponente kann Mausbotschaften empfangen
  h) Eine Komponente kann Tastaturbotschaften empfangen
  i) Eine Komponente kann mit der virtuellen Maschinen von Zemprosys kommunizieren
  j) Eine Komponente kann indirekt die System- bzw. Betriebssystemfunktionen aufrufen
  k) Eine Komponente kann direkt und indirekt alle Delphi-Compilerfunktionen aufrufen

## 6.4. ERWEITERUNG DER KOMPONENTENBIBLIOTHEK

Das wesentliche Element der Architektur von Zemprosys ist die Komponentenbibliothek. Um bedeutende Veränderungen der Software vorzunehmen, muss man daher mehr Akzente auf die Komponentenbibliothek setzen. Die Erweiterung dieser Bibliothek durch neue Komponenten, Methoden, Aktionen, Schnittstellen und Botschaften unterliegt einem exakten Plan. Bei der Planung müssen diverse Technologie-Aspekte berücksichtigen würden. Die Anwendungsbereiche von Komponenten sind dabei sehr wichtig, damit die Software gezielt weiterentwickeln wird. Auch die Multifunktionalität von Komponenten repräsentiert einen wichtigen Planungsfaktor.

Damit die Basistechnologie erweitert und verbessert wird, muss man zunächst feststellen, wie umfangreicher die Komponentenbibliothek ist. Der Ist-Zustand dieser Bibliothek muss qualitativ und quantitativ ermittelt werden. Anhand der Ist-Zustand kann man dann eine Bedarfsliste erstellen. Eine optimale Entwicklungsplanung im Kontext von Zemprosys setzt mindestens folgende Schritte voraus:

### (1) Erstellung einer Komponententabelle

Alle Komponenten, die entwickelt werden müssen, werden in einer Tabelle zusammengefasst. Diese Tabelle besitzt folgende Spalten:

- **Komponentenname**. Jede Komponente wird über einen Name angesprochen. Bei der Festlegung des Namens, muss man zunächst prüfen, welche Komponentennamen schon registriert sind, weil duplizierte Namen nicht erlaubt sind.
- **Komponenten-Identifikation**. Jede Komponente besitzt eine Identifikationsnummer, die eindeutig sein muss. Die Vergabe von Nummern (MyIden) unterliegt ebenfalls bestimmte Regeln.
- **Anwendung**. Zusammenfassung des Anwendungsbereiches. Allgemeine Aufgaben der Komponente werden beschrieben. Dadurch wird festgestellt, ob eine ähnliche Komponente schon existiert und wenn ja, prüft man, ob es besser ist, diese existierende Komponente zu erweitern (siehe auch Status).
- **Status**. Bestimmt den Ausgangspunkt der Entwicklung und kann folgende Werte annehmen: Komponente ist neu, Komponente ist ein Nachkommen, Komponente wird erweitert.
- **Komponentengruppe**. Die Komponentenbibliothek ist in Gruppen aufgeteilt. Beispielsweise existieren die Gruppen Standard, Dokumente und Bilder.

- **Verfügbarkeit**. Legt fest, ob die Komponente von Benutzern genutzt werden darf oder nicht. In der Regel bei der Festlegung der Komponentengruppe bestimmt man auch zugleich den Wert der Verfügbarkeit.
- **Vererbung**. Ist die Komponente ein Nachkommen, muss der Name des Vorfahrs eingegeben werden.
- **Multifunktionalität**. Zusammenfassung der anderen möglichen Anwendungsbereiche, die über die Standardanwendungen hinaus von der Komponente erfüllt werden.

## (2) Erstellung einer Eigenschaftstabelle

Die Aufgabe einer Eigenschaftstabelle ist die Auflistung von Eigenschaften, die während der Entwurfszeit vom Benutzer konfiguriert werden können. Die Spalten der Tabelle sind:

- **Eigenschaftsname**. Muss innerhalb des Komponentenbereiches eindeutig sein.
- **Eigenschaftstyp**. Wie Farbe, Text, Zahl usw. Bei der Festlegung des Types bedient man sich von Zemprosys-Datentypen.
- **Eigenschaftsgruppe**. Eigenschaften einer Komponenten können in Gruppen aufgeteilt werden, wobei die selbe Eigenschaft in mehr als eine Gruppe eingetragen werden kann.
- **Eigenschaftsaufgabe**. Kurzbeschreibung der Aufgabe. Wenn innerhalb des Komponentenbereiches eine solche Eigenschaft schon existiert, ist es besser die Eigenschaft nicht in der Tabelle einzutragen.

## (3) Erstellung einer Methodentabelle

Die Aufgabe einer Methodentabelle ist die Auflistung von Operationen, die von einer Komponente während der Entwurfszeit ausgeführt werden können. Die Spalten der Tabelle sind:

- **Methodenname**. Muss ebenfalls innerhalb des Komponentenbereiches eindeutig sein.
- **Methodentyp**. Ist entweder eine Lokalmenüoperation oder eine Inspektor-Operation. Eine Inspektoroperation wird wie eine Eigenschaft den Benutzern zur Verfügung gestellt.
- **Methodengruppe**. Methoden für Inspektor-Operationen können in Gruppen aufgeteilt werden, wobei die gleiche Methode in mehr als eine Gruppe eingetragen werden kann.
- **Methodenparameter**. Methodenparameter sind nur bei den Inspektor-Operationen möglich. Die Parameter werden während der Ausführung der Operation übergeben.
- **Methodenaufgabe**. Kurzbeschreibung der Aufgabe. Wenn innerhalb des Komponentenbereiches eine solche Methode schon existiert, ist es besser die Methode nicht in der Tabelle einzutragen.

## (4) Erstellung einer Aktionstabelle

Die Aufgabe einer Aktionstabelle ist die Auflistung von Aktionen, die von einer Komponente während der Laufzeit ausgeführt werden können. Aktionen bilden den Anfang der Eventprogrammierung. Die Spalten der Tabelle sind:

- **Aktionsname**. Muss ebenfalls innerhalb des Komponentenbereiches eindeutig sein.
- **Aktionsparameter**. Parameter, die während der Ausführung der Aktion übergeben werden.
- **Aktionsaufgabe**. Kurzbeschreibung der Aufgabe. Wenn innerhalb des Komponentenbereiches eine solche Aktion schon existiert, ist es besser die Aktion nicht in der Tabelle einzutragen.

## (5) Erstellung einer Schnittstellentabelle

Die Aufgabe einer Schnittstellentabelle ist die Auflistung von Operationen, die von einer Komponente ausgeführt werden können und die zu Kommunikation mit der Außenwelt (externe Systeme) dienen. Es kommt öfter vor, dass eine Aktion zugleich eine Schnittstelle ist oder umgekehrt. Die Spalten der Tabelle sind:

- **Schnittstellenname**. Muss ebenfalls innerhalb des Komponentenbereiches eindeutig sein.
- **Schnittstellenparameter**. Parameter, die während der Aktivierung der Schnittstellen übergeben werden.
- **Schnittstellenverfügbarkeit**. Bestimmt, ob die Ausführung der Schnittstelle eingeschränkt ist oder nicht. Bei einer eingeschränkten Ausführung, dürfen nur Nachkommen der Komponente diese Schnittstelle aufrufen.

- **Schnittstellenaufgabe**. Kurzbeschreibung der Aufgabe. Wenn innerhalb des Komponentenbereiches eine solche Schnittstelle schon existiert, ist es besser die Schnittstelle nicht in der Tabelle einzutragen.

### (6) Erstellung einer Botschaftstabelle

Die Aufgabe einer Botschaftstabelle ist die Auflistung von Botschaften, die von einer Komponente gesendet werden können und die zu Kommunikation mit der Außenwelt (externe Systeme) dienen. Das Senden von Botschaften ist ein praktisches Kommunikationsinstrument. Die Spalten der Tabelle sind:

- **Botschaftsname**. Muss ebenfalls innerhalb des Komponentenbereiches eindeutig sein.
- **Botschaftsparameter**. Parameter, die während der Botschaftssendung übergeben werden.
- **Botschaftsaufgabe**. Kurzbeschreibung der Aufgabe. Wenn innerhalb des Komponentenbereiches eine solche Botschaft schon existiert, ist es besser die Botschaft nicht in der Tabelle einzutragen.

**Fazit:**
Die beschriebene Methodik für die Entwicklungsplanung ähnelt der praktischen Vorgehensweise bei der Softwareentwicklung und bei Objektorientierter Programmierung. Anhand des erstellten Plans und auf Basis von Zemprosys-Architekturkonzept, wird dann die Erweiterung der Komponentenbibliothek konkretisiert. Modellierungs- und Programmieraufgaben werden unter einem vorgegebenen Design durchgeführt.

## 6.5. SOFTWARE-VERBESSERUNG

Was bis jetzt Zemprosys auszeichnet, ist das Design der Benutzerschnittstelle. Dieser Benutzerschnittstelle ist innovativ und setzt einen Trend ein. Aber das Ziel „Entwicklung von Computerprogrammen ohne Code-Schreibung“ setzt voraus, dass eine intelligente und intuitive Benutzerschnittstelle den Nutzern angeboten wird. Die aktuelle Benutzerschnittstelle zu Durchführung der visuellen Programmierung ist schon anspruchsvoll und innovativ, aber eine Verbesserung muss vorgenommen werden. Für die Verbesserung der Bedienoberfläche soll die Software mit folgenden Funktionen erweitert werden:

### (1) Individualisierungssystem zur Optimierung der Arbeit

Durch die Implementierung eines Individualisierungssystems werden den Benutzern ein Instrument zur Verfügung gestellt, um eine Arbeitspersonalisierung vorzunehmen. Einige Aspekte der Personalisierung sind:

- Speichern und Wiederladen eines Desktopzustandes
- Deaktivierung und Wiederaktivierung von nicht häufig benutzten Funktionen
- Favoriten bei der Auswahl von Komponenten setzen
- Intelligente Einstellung von Komponenten (Einfacher Zugriff, Favoriten für Eigenschaften und Methoden einer Komponente setzen, Komponentenvorgaben speichern und wieder laden)
- Setzen von Referenzen bzw. Shortcuts
- Versionierung der Arbeit (CMS für Projektdateien und Komponenten)
- Benutzerverwaltung für Projekte (Passwort, Änderungsrechte, Leserechte, Benutzergruppe)

### (2) Intuitives Lernsystem für die Benutzung von Komponenten

Ein der Nachteile des VPPT-Verfahrens ist diese Voraussetzung, dass der Benutzer die Funktionen und Elemente der Technologie richtig kennen muss, um leistungsfähige Applikationen erzeugen zu können. Die Zemprosys-Logik besagt, dass der Benutzer die Interpretation jeder Komponenteneigenschaft oder jede Komponentenmethode richtig einordnen muss, um sie auch effektiv einsetzen zu können.
Obwohl bei der Änderung einer Komponenteneigenschaft oder bei der Ausführung einer Komponentenmethode die Änderungsergebnisse sofort visualisierbar sind, ist es besser im Voraus zu

wissen, welche Funktionalitäten besitzen die Zemprosys-Elemente. Das folgende Bild zeigt beispielsweise, welches Problem der Benutzer haben wird, wenn er keine gezielte Informationen über die zusehenden Begriffe erhält:

**Abbildung 5:** Zemprosys-Elemente als Checkliste

Zum Erläutern dieses Bildes soll der Benutzer eine Komponente einstellen, indem er einige Optionen dieser Komponente aktiviert oder deaktiviert. **QInfo_AutoSizeTopic** ist beispielsweise ein Zemprosys-Element, das beschrieben werden muss.
(Im Softwarekontext führt die Einstellung bzw. Aktivierung von **QInfo_AutoSizeTopic** dazu, dass die Höhe eines Themen-Bereiches der entsprechenden Komponente automatisch berechnet wird)

Die Implementierung eines intuitiven Lernsystems ist daher notwendig. Ein solches Lernsystem muss folgende Funktionen anbieten:

- Automatische und gezielte Hilfebereitstellung
- Manuelle Hilfebereitstellung nach Anforderung des Benutzers
- Hilfe durch Illustrationen, wenn dies erforderlich ist
- Beschreibung von Alternativen und Vergleichung von Elementen

## (3) Dialogsystem für eine optimale Konfiguration von Komponenten

Das Problem, das gerade angesprochen wurde (siehe Punkt (2)), soll durch eine alternative Designlösung behoben werden. Zwar werden die Zemprosys-Elemente dadurch nicht an Bedeutung verlieren, aber der Benutzer soll in eine Dialogsituation versetzt werden, während er gerade ist, eine Komponente zu konfigurieren. Die Funktionen des Dialogsystems sind unter anderem:

- Die Zemprosys-Elemente, also solche Begriffe wie QInfo_AutoSizeTopic werden in bekanntlichen Sprachen wie Englisch automatisch übersetzt
- Bereitstellung eines Konfigurationsschemas für alle Komponententype
- Das Konfigurationsschema arbeitet überwiegend mit Illustrationen und Vorschau
- Das Konfigurationsschema verfügt über eine Navigationsfunktion für „vorwärts konfigurieren“, „rückwärts konfigurieren“ und „Konfigurationskapitel wählen“
- Zusätzliche Navigationshilfe

## (4) Frontend und Backend für Komponenten

Zum Zweck der Benutzerfreundlichkeit, Minimierung der Entwicklungszeit und Steigerung der Effektivität wird der Zugriff auf Komponenten mit einer zusätzlichen Alternative (Frontend-Backend) unterstützt. Das Frontend-System erlaubt den Zugriff auf solche Eigenschaften und Methoden, die überwiegend eine Auswirkung auf die Visualisierung und GUI haben. Das Backend-System beschäftigt sich mehr mit solchen Eigenschaften und Methoden, die zwar eine Hintergrundarbeit leisten, die aber sehr wichtig sind. Die Eventcodierung und Eventspeicherung gehören beispielsweise zum Backend-System.

## (5) Texteditor zur manuellen Bearbeitung von Projektdateien

Es hat sich herausgestellt, dass ein reiner Texteditor für bestimmte Phasen des VPPT-Verfahrens geeignet ist, um die Änderungen an Projektdateien unkompliziert vorzunehmen. In einigen Fällen lassen sich Komponentenkonfigurationen besser durch Texteingabe steuern.

### (6) Drucken von Zemprosys-Inhalten

Inhalte von Zemprosys lassen sich momentan nur sehr schwer drucken, wegen der starken Ähnlichkeit zwischen Dokumenten und Applikationen. Zemprosys-Dokumente sind dynamisch und verhalten sie sich in manchen Fällen wie Computerprogramme. Die Aufgabe besteht daher darin, ein unkompliziertes und intelligentes Druckverfahren zu entwickeln, das eine Zemprosys-Applikation von einem Zemprosys-Dokument unterscheiden kann. Wenn beispielsweise ein Dokument über Schaltflächen verfügt, soll das Druckverfahren in der Lage sein, nur reine Inhalte zu drucken oder selektiv bestimmte Informationen als Druckpaket zu bilden.

## 6.6. EDITIONEN

Ein Technologie-Trick mit dem Codename „EDITION“ wird von Zemprosys angewendet, um mehrere eigenständigen Computerprogramme von der Festplatte starten zu lassen, indem nur ein einziges Programmfenster für alle dieser Programme verwendet wird. Das Wort EDITION ist in diesem Fall identisch mit einem Computerprogramm.
Während des Startprozesses eines Zemprosys-Programms (z.B. ZDS) können mehrere EDITIONEN gestartet werden, die betriebssystemunabhängig laufen können. Der Trick dabei ist die Verwaltung von EDITIONEN über die virtuelle Maschine von Zemprosys (Herz der Technologie). Statt dieser EDITIONEN vom Betriebssystem verwalten zu lassen, übernimmt die virtuelle Maschine die komplette Steuerung dieser spezifischen Programme. In diesem Verfahren ist nur die virtuelle Maschine abhängig vom Betriebssystem und EDITIONEN bleiben unbekannt für das Betriebssystem.

EDITIONEN bringen viele Vorteile mit sich, sowohl technisch als auch Kommerziell:

a) Leichte Kommunikation zwischen EDITIONEN bzw. einzelnen Programmen
b) Verschiedene EDITIONEN können Zugriff auf gleiche Ressourcen, die sich im Speicher befinden, haben
c) Leichte Zentralisierung und Dezentralisierung der Aufgaben
d) Die Erweiterung der Technologie ist leicht durchführbar und kundenspezifische Funktionen können ebenfalls leicht eingebaut werden
e) Starke Unterstützung des VPPT-Verfahrens
f) Vereinfachte Forme bei der Lizenzvergabe
g) Benutzerfreundlich und Übersichtlichkeit, da nur ein einziges Programmfenster für verschiedene Programme zur Verfügung steht.

Einige EDITIONEN sorgen dafür, dass bestimmte Aufgaben, die zum Teil sehr komplex sind, optimal und problembezogen gelöst werden. Diese EDITIONEN sind teilweise entwickelt worden und sollen weiter verbessert werden:

- **People Edition**. Bedienoberfläche zur Durchführung der visuellen Programmierung
- **OpenOddic ZM**. Integrierter Klassen-Analyzer für wichtige Aufgaben des Systems. OpenOddic dient beispielsweise zu Sicherstellung der Multisprache für Überschriften, Texte, Systemmeldungen und Hilfe
- **People project**. Pejektverwaltungsprogramm mit integriertem CMS (Content Management System) und DMS (Document Management System)
- **Code Editex ZM**. Automatisierungsprogramm zur Beschleunigung der Komponentenentwicklung und zum Dokumentieren von Zemprosys

## 6.7. OPEN ZEMPROSYS

OpenZemprosys ist ein interner Begriff der Technologie und darf nicht mit Open Source verwechselt werden. OpenZemprosys soll als System verstanden werden und wird als erstes während des Startes eines Zemprosys-Programms (z.B. ZDS) eingerichtet. Dieses System dient dann als zentrale

Schnittstelle des Benutzers. Auch wenn die Entwicklung dieses Systems noch in seiner Frühphase steht, ist seine Aufgabe dennoch sehr relevant.

OpenZemprosys ist für Benutzer eine Standardoberfläche für die Zentralisierung von verschiedenen Aufgaben, die unter anderem aus folgenden Punkten bestehen:

- Direkter Zugriff auf alle registrierten Anwendungen (EDITIONEN) von Zemprosys
- Benutzerschnittstelle für Netzwerk- und Webapplikationen (MyZemprosys). Alle autorisierten Funktionen und Applikationen für den Nutzer werden durch MyZemprosys aktiviert und verwaltet
- Beim Starten von Zemprosys werden alle registrierten Projekte ermittelt und aufgelistet. Der Anwender hat dadurch die Möglichkeit, Projekte direkt ohne „Durchsuchen“ zu öffnen. Um Projekte zu lokalisieren, sucht Zemprosys automatisch anhand von bestimmten Einstellungen nach Projekten in Systemverzeichnissen
- Durchführung von Softwareeinstellungen
- Ein- und Auslogen-Funktionen
- Bereitstellung diverser Webservice durch Turbo Informer (Emailservice, Abrufen von Internetseiten, Newsletter, Teamarbeit durch E-Kollaboration usw.)

## 6.8. PORTIERUNG DER TECHNOLOGIE

Bei der Portierung der Technologie geht es mehr darum ein **Cross System** zu entwickeln, damit die VPPT-Applikationen auf verschiedene Systeme laufen. Insbesondere soll verschiedene Versionen des Laufzeitprogramms ZCI geben, damit die Windows-, Linux- und Mac-Benutzer zufrieden gestellt werden.

Bei der Entwicklung eines Cross Systems sind folgende Ansatzpunkte für ein tragfähiges Protierungskonzepts zu berücksichtigen:

a) Zemprosys hat eine Architektur mit vorgegebenen Regeln. Diese Architektur wird systematisch für eine systemunabhängige Portierung als Basis verwendet.

b) Zemprosys verfügt über eine komponentenbasierte Architektur und unterstützt somit die objektorientierte Programmierung. Alle objektorientierten Programmiersprachen haben einen gewissen gemeinsamen Nenner. Man kann somit von Zemprosys die nötigen Abbildungen für Java und C/C++ generieren, die dann für die Portierung dienen.

c) Zemprosys wird zwar mit Hilfe des **Borland Delphi Compilers** entwickelt, aber die Architektur ist fast 90% unabhängig von Delphi-Ressourcen (Komponenten), weil alle Zemprosys-Komponenten neu entwickelt wurden. Auch die Windows-Appi-Funktionen werden im begrenzten Umfang verwendet.

d) Eine totale Portierung erfolgt, indem Zemprosys mit Borland Delphi Compiler weiter entwickelt wird und der Source Code wird gleichzeitig in den Programmiersprachen Java und C/C++ konvertiert. Sofern also der Source Code von Zemprosys in Delphi, Java und in C/C++ vorliegt, kann die Technologie plattformunabhängig laufen (Windows, Linux, Mac)

### (1) Partielle Portierung

Durch Bereitstellung von ZCI (Zemaurice Code Interpreter) für verschiedene Betriebssysteme wird eine partielle Portierung insofern erreicht, weil die VPPT-Applikationen in Textdatei (Skript-Form) geschrieben werden können –und dann von ZCI unabhängig von Plattformen korrekt ausgeführt werden.
Die Aufgabe der partiellen Portierung liegt in der Entwicklung von ZCI für die Systeme Linux und Mac. Dementsprechend soll auch ZDS (Zemaurice Devoloper Studio) portiert werden.

### (2) Totale Portierung

Während bei der partiellen Portierung den Source Code nur in Object Pascal Form vorliegt, handelt es sich bei einer totalen Portierung um die Konvertierung des Object Pascal Code in Java und C/C++.
Zur Erreichung der totalen Portierung werden zwei Automatisierungsprogramme entwickelt, die den Source Code von Zemprosys in Java und C/C++ automatisch konvertiert, jedes mal wenn Zemprosys verbessert und aktualisiert wird:

- ZemprosysToJava konvertiert Zemprosys(Delphi) in Zemprosys(Java)
- ZemprosysToC/C++ konvertiert Zemprosys(Delphi) in Zemprosys(C/C++)

# 7. ZEMPROSYS INTRO

## 7.1. ÜBERBLICK

Zemprosys Intro ist ein Tool für Web- und Intranetanwendungen mit zwei wichtigen Elementen: Die Verknüpfung von MySQL Server als Backend-System und Zemprosys als Frontend-System. Von diesem Backend-Fontend-System wird Zemprosys Intro abgeleitet (siehe Kap. „Technologie“)

Die Entwicklung von Zemprosys Intro steht am Anfang. Dennoch ist die Grundidee dieses Produkts revolutionär, weil der gesamte Code des Tools in Form von Skript-Applikationen über eine relationale Datenbank (MySQL) verfügbar ist. Zemprosys Intro ist im Grunde eine Datenbank bestehend aus Applikationen, die durch SQL-Anweisungen ausgeführt werden können. Vom Prinzip her braucht der Benutzer das Laufzeitprogramm (ZCI), das in diesem Zusammenhang als Clientsoftware fungiert und das die Anfragen an MySQL Server startet. Antworten auf SQL-Anfragen sind meistens Skript-Applikationen (Sammlung von Softwarekomponenten mit bestimmten Zustände), die von ZCI interpretiert und auf dem Bildschirm angezeigt werden.

## 7.2. TOOL BESCHREIBUNG

Zemprosys Intro ist eine Datenbank bestehend aus folgenden einzelnen Systemen:

- Worker Interface System
- Project Management System
- Reporting System
- Berechtigungssystem
- Messaging System
- Meeting System
- Document Management System (DMS)
- Community System
- Administration System
- Clean Box System
- Open Directory System
- Project Base Solution
- VPPT Directory System
- Worker Creating Interface

### - Worker Interface System

Dieses System definiert den virtuellen Arbeitsplatz eines Clients, der Zemprosys Intro nutzt. Als Clients kommen beispielsweise Mitglieder eines Projektteams oder Mitarbeiter einer Firma in Frage. Wird die Anwendung auf einen Webserver ausgeführt, kann ein User überall hier (ortsunabhängig und zeitsunabhängig) die Funktionen dieses Systems ausführen.

## - Project Management System
Das System deckt die Grundfunktionen üblicher Projektmanagement-Tools.

## - Reporting System
Für das Berichtswesen. Dieses System stellt beispielsweise Funktionen bereit, um Auswertungen zu definieren, nicht nur aus dem Projektmanagement, sondern aus allen Bereichen.

## - Berechtigungssystem
Mitarbeiter oder Projektmitglieder haben durch ein Berechtigungsverfahren Zugriff auf Ressourcen. Das Berechtigungssystem darf nur von Systemadministratoren und Co-Systemadministratoren benutzt werden: Co-Systemadministratoren unterliegen dem Systemadministrator. Funktional betrachtet ist der Funktionsumfang von Co-Systemadministratoren im Vergleich zu Systemadministrator geringer.

## - Messaging System
Das Messaging System gewährleistet die elektronische bzw. virtuelle Kommunikation innerhalb eines Projektteams oder eines Unternehmens (Direkte Mitteilungen und Kommentierungen zu Dokumenten, Vorgängen usw.)

## - Meeting System
Dieses System übernimmt die Steuerung von virtuellen Meetings. Wenn man das Meeting System auf Unternehmensebene betrachtet, handelt es sich um eine Momentaufnahme: Alle Beteiligten sollen sich virtuell zu einem bestimmten Zeitpunkt vereinbaren, um eine Live-Diskussion durchzuführen (ist nicht zeitversetzt). Mit dem Messaging System haben die Beteiligten ein relatives großes Zeithorizont für Reaktionen auf Mitteilungen, Kommentierungen usw. (Reaktionen können zeitversetzt sein)

## - Document Management System (DMS)
Verwaltung von Dokumenten und Archiven

## - Community System
Dieses System kann als allgemeiner Browser für registrierte Benutzer verstanden werden und stellt eine Reihe nützlicher Funktionen zur Verfügung. Beispielsweise die Bildung von Netzwerken innerhalb eines Unternehmens.

## - Administration System
Dieses System stellt Administrationsfunktionen zur Verfügung, die nur für Systemadministratoren reserviert sind.

## - Clean Box System
Papierkorb /Verwaltung und Wiederherstellung von gelöschten Informationen.

## - Open Directory System
Für Knowledge: Ideen, Lösungsvorschläge etc. werden unabhängig von Projekten und Abteilungen zentral gemanagt. Es handelt sich hier um ein Wissenspool.

## - Project Base Solution
Die schwache Stelle der gewöhnlichen Projektmanagement-Tools ist teilweise dadurch kenngezeichnet, dass diese Tools keine große Flexibilität für die Verarbeitung und Visualisierung von Daten anbieten. Sobald die Projekte komplex werden, weisen die Projektmanagement-Tools Grenze bei der Darstellung von Informationen auf. Mit „Project Base Solution“ soll nun ein Verfahren basierend auf VPPT den Nutzern bereitgestellt werden, damit Projektmodelle generiert

werden. Insbesondere für komplexe Projekte etwas wie in der Produktion soll „Project Base Solution“ ihre Stärke spielen.

Die Idee hinter "Project Base Solution" ist die: Der Systemadministrator und die Projektleiter sollen die Möglichkeit haben, ein maßgeschneidertes Projektmanagement-System selbst zu entwickeln, wenn das Standard für bestimmte Projekte nicht mehr ausreicht. Dies ist insbesondere wichtig, weil einige Projekte, die aus Realisierungssicht her sehr komplex sind, durch eine Standard-Vorgehensweise nicht mehr optimal gesteuert werden.

### - VPPT Directory System

Dieses System verwaltet die VPPT-Applikationen (Skript-Applikationen), die spezielle Lösungen für Kunden abdecken. Also, Zemprosys Intro kann auch mit maßgeschneiderten Lösungen ausgebaut werden.

### - Worker Creating Interface

Es ist ein Werkzeug, das eine Dynamisierung von *Worker Interface System* ermöglicht. Mit dem Werkzeug haben die Benutzer von Zemprosys Intro die Mittel, ihre virtuellen Arbeitsplätze mit zusätzlichen Funktionen zu bereichern und somit eine Hoch-Personalisierung durchzusetzen. Diese Benutzerschnittstelle nutzt das VPPT-Verfahren, also die visuelle Programmierung durch Benutzer, um die Implementierung einer Hoch-Personalisierung zu unterstützen.

## 7.3. ENTWICKLUNGSPAKET

### (1) Minimale Anforderung

Die minimale Funktionalität zur Implementierung von Zemprosys Intro ist die Schnittstelle zu MySQL Server, die die Ausführung von SQL-Anweisung möglich macht. Diese Schnittstelle soll unter anderem über die folgenden Mechanismen verfügen:

- Schreiben von Zemprosys Projektinformationen in einer MySQL Datenbank
- Lesen von Zemprosys Projektinformationen aus einer MySQL Datenbank
- Schreiben von Skript-Applikationen in einer MySQL Datenbank
- Lesen von Skript-Applikation aus einer MySQL Datenbank
- Erzeugung von Datenbanken und deren Tabellen
- Verwaltung von Datenbanken und deren Tabellen

Bei der Sicherstellung der minimalen Funktionalität geht es generell darum, das Backend-System zu entwickeln. Neben der Anforderung zur Verarbeitung von SQL-Anweisungen soll das Backend-System ferner folgende grundlegende Eigenschaften aufweisen:

- Einfache Netzwerkadministration
- Strikte Trennung zwischen Daten und Applikationen. Ein sehr wichtiger Aspekt, weil die Unternehmensdaten von Zemprosys Intro (auch eine Datenbank) abgetrennt werden müssen
- Transparenz, Sicherheit und Zuverlässigkeit (Merkmale von MySQL Server, die ohne weiteres garantiert werden müssen)
- Unkomplizierte Migration von Daten (MySQL Organisation bieten schon einige Tools, die diese Aufgabe problemlos erfüllen können)
- Skalierbarkeit von Zemprosys Intro, indem einzelne Systeme (Skript-Applikationen) hinzugefügt oder entfernt werden können, ohne das Gesamtsystem zu destabilisieren
- Personalisierung und Berechtigung. Der Zugriff auf Zemprosys Intro soll durch ein Benutzerverwaltungsverfahren basiert auf Personalisierung und Berechtigung erfolgen. Diese Aufgabe stellt die Grundlagen des Berechtigungssystems von Zemprosys Intro dar
- Versionierung von Zemprosys-Projekten

## (2) Minimale Aufgabe /Arbeitspakete

Die erste Stufe der Entwicklung von Zemprosys Intro umfasst folgende Arbeitspakete:

### Phase 1: Entwicklung des Backend-Systems

Die oben beschriebene minimale Anforderung soll bei der Durchführung dieses Arbeitspakets erfüllt werden.

- Konzeption von Zemprosys Intro (Entwerfen von Tabellenstrukturen und Verknüpfungen)
- Programmierung der Schnittstelle zu MySQL Server

### Phase 2: Entwicklung eines Backend-Editors

Der Editor soll der Zugriff auf das Backend-System durch Automatismen und visuelle Elemente vereinfachen

- Schreiben und Lesen von Zemprosys Projektinformationen
- Schreiben und Lesen von VPPT-Applikationen
- Versionierung von Zemprosys Projekten
- Benutzerverwaltung

### Phase 3: Entwicklung eines integrierten MySQL-Datenbankeditors

Mit dem Editor soll die Entwicklung von MySQL-Datenbanken über die Plattform ZDS vereinfacht werden

- Anlegen von Datenbanken und deren Tabellen
- Verwaltung von Datenbanken und deren Tabellen
- Verarbeitung von SQL-Anfragen

### Phase 4: Entwicklung eines integrierten MySQL-Datenbankbrowsers

- U.a. Visualisierung, Ändern, Löschen, Einfügen, Kopieren, Suchen, Sortieren und Gruppieren von Datensätzen
- Explorieren von Tabellen, Vorwärts Anzeige, Rückwärts Anzeige usw.

### Phase 5: Entwicklung eines Layouts für das Frontend-System

Für alle im Abschnitt „Definition“ aufgelisteten einzelnen Systeme von Zemprosys Intro ist ein Markenzeichen zu erschaffen, mit dem sich das Produkt in Zukunft identifizieren wird

- Konzeption einer einheitlichen Benutzerschnittstelle (GUI) für alle einzelnen Systeme
- Beschaffung von systemübergreifenden Grafikelementen (Bilder, Icons und Schaltfläche) für benutzerfreundliche Bedienoberflächen
- Definieren der allgemeingültigen Grundfunktionen

### Phase 6: Entwicklung einer Standard-Basiskomponente

Die Konzeptionsarbeit des Arbeitspakets 5 wird übernommen und durch Programmierung eine visuelle Basiskomponente umgesetzt.

- GUI Programmierung
- Programmierung der Schnittstelle zu Backend-Systems

### Phase 7: Erstellung von Zemprosys Intro ohne Funktionen und Daten

Auf Basis der Ergebnisse des Arbeitspakets 6 wird Zemprosys Intro - wie andere Zemprosys Projekte - angelegt und mit Hilfe des Backend-Editors in MySQL-Datenbank konvertiert. In diesem Entwicklungsstadium ist Zemprosys Intro lediglich nur ein leeres System, weil die Skript-Applikationen, die eigentlich das System ausmachen, noch nicht programmiert worden sind.

## (3) Weiterführende Aspekte

Wenn man das Konzept von Zemprosys Intro mit den aktuellen Produkten am Markt vergleicht, kann es leicht verstanden werden. Bei der Produktion von Webseiten und meisten CMS wird schon ein fast ähnliches Konzept angewendet. Mit Typo 3, 4 beispielsweise werden die Inhalte und Parameter

n einer Webseite in einer relationalen Datenbank gespeichert und die stehen auch in dieser Form weiterhin zu Verfügung.

Sobald das Backend-System - das eine komplexe Sache ist - entwickelt worden ist, ist alles möglich und Dank der Skript-Applikationen. Viele Anwendungsgebiete können dann an das System integriert werden. Die Skalierbarkeit von Applikationen ist auf jedenfalls gewährleistet ohne die Unternehmensdaten zu gefährdet.

Bis zum Arbeitspaket 7 ist man mit einer rudimentären bzw. klassischen Programmierung beschäftigt. Aber für die zweite Stufe der Entwicklung werden einige Aufgaben mit ZDS erledigt: Die Programmierung der einzelnen Systeme erfolgt nur noch visuell durch VPPT. Dies bedeutet aber nicht, dass die klassische Programmierung nicht mehr durchgeführt wird, weil beispielsweise geeignete Events noch eingebaut werden müssen oder einige Type von Komponenten neu entwickelt werden müssen.

# 8. CLIENT/SERVER-MODELL

## 8.1. ÜBERBLICK

Das Client/Server-Modell „Centerix“ ist nur ein Konzept auf dem Papier, dessen Umsetzung nicht unbedingt heute oder morgen vorgenommen werden muss. Dennoch haben wir im Blick, eine eigene Client/Server-Technologie auf den Markt zu bringen. Während Zemprosys Intro total auf MySQL Server aufbaut, soll Centerix mehr Unabhängigkeit von etablierten Datenbanktechnologie gewährleisten. Zumindest soll der Benutzer bei der Wahl von Datenbankverwaltungssystem wie z.B. MySQL mehrere Alternativen haben.

Centerix kombiniert die Vorteile des VPPT-Verfahrens mit Client/Server-Funktionen, um eine vorgegebene Netzwerktechnologie zu realisieren:

- Das VPPT-Verfahren wird als zentrales Realisierungsinstrument eingesetzt, indem die VPPT-Applikationen vom Server aufgerufen und ausgeführt werden.

- Benutzer von Centerix brauchen die Software ZCI (Zemaurice Code Interpreter), die als Client-Software agiert und die Kommunikation zwischen den Clients und dem Server steuert. ZCI wird standardmäßig kostenlos vertrieben.

- Obwohl Centerix die Webtechnologie bzw. die Kommunikation durch Internet aufbaut, ist diese Client/Server-Architektur nicht total webbasiert, weil die VPPT-Applikationen von einem Internetbrowser nicht ausgeführt werden können. Das Ziel ist zumindest nicht die Konvertierung von VPPT-Applikationen in HTML oder in sonstigen Internetformaten. Das Aufrufen und die Ausführung von VPPT-Applikationen wird hauptsächlich von der Client-Software ZCI sichergestellt.

- Die Innovation mit Centerix besteht darin, dass das Softwaresystem eines Unternehmens in Textdateiformat (Skript-Applikationen) gebildet und erzeugt werden kann. Dies erleichtert die Speicherung, minimiert den Administrationsaufwand, optimiert und beschleunigt den Zugriff auf Benutzerapplikationen. Komplexe Installationen kommen nur selten in Frage.

## 8.2. DESIGN

Die nächste Graphik bzw. das nächste Bild ist eine Abbildung der Architektur von Centerix mit folgenden Elementen:

**VPPT CLIENT**. Ein VPPT CLIENT ist ein Benutzer, der mit ZTS mit Hilfe einem PC , Thin-Client oder einem Laptop kommuniziert, um die Netzwerkfunktionen und Netzwerkressourcen aufzurufen. Die dafür benötigte Client-Software ist ZCI. ZCI ruft die entsprechenden VPPT-Applikationen auf und führt sie aus.

**ZTS (ZEMPROSYS TRANSACTION SERVER).** ZTS ist ein Serverprogramm für Transaktionen und hat die Aufgabe, Anfragen von Clients und externen Anwendern zu verarbeiten. ZTS definiert das Frontend von Centerix.

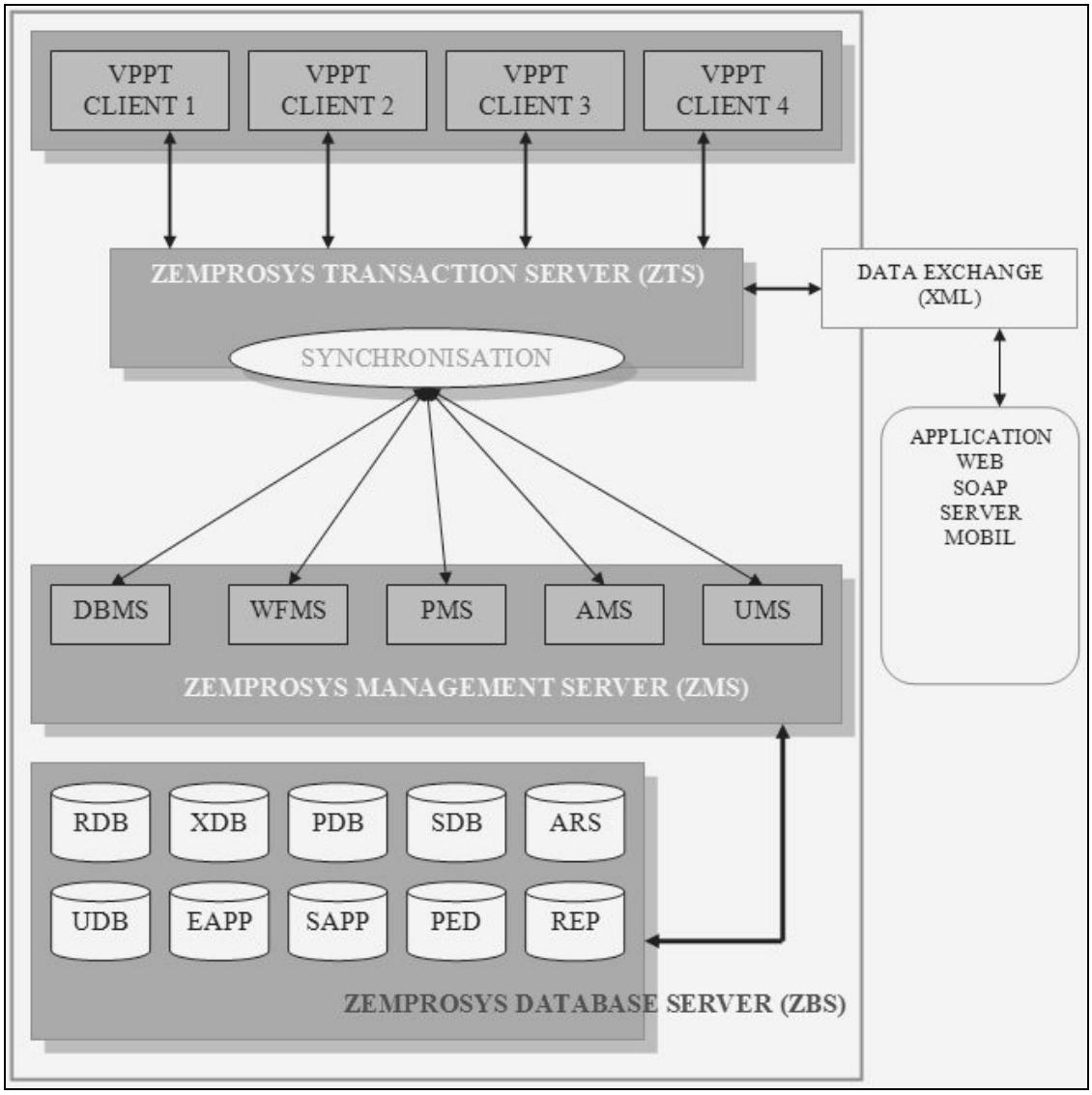



**Abbildung 6:** Architektur von Centerix

**ZMS (ZEMPROSYS MANAGEMENT SERVER).** Der Management-Server übernimmt die Organisation des Systems und unterteilt es in folgenden Bereichen:

- **DBMS** (Database Management System)

- **WFMS** (Workflow Management System)

• **PMS** (Project Management System). PMS übernimmt die Verwaltung von Projekten. Verschiedene Projekttype können über Centerix durchgeführt werden. Die Menge von Projekttypen ist eine dynamische Größe, weil deren Anzahl mit der Zeit und je nach Anforderung wachsen kann. Bei der Durchführung von Projekten bekommen die Benutzer bzw. Projektmitglieder die erforderlichen Werkzeuge und Softwarekomponenten. Konzeptionell betrachtet, sind Projekttype vorgegebene Arbeitsmodelle, die auf Basis des VPPT-Verfahrens entworfen werden. Beispiele eines Projekttyps ist die Prozessvisualisierung im Industriebereich.

• **AMS** (Application Management System). Verwaltung von VPPT-Applikationen, die den Benutzern zur Verfügung gestellt werden. Eine der wichtigen Aufgaben von AMS ist die Umsetzung der Zugriffsberechtigung, indem die Clients nur auf erlaubte Applikationen zugreifen. VPPT-Applikationen werden in XML-Format oder in ZemprosysScript gespeichert und an den Clients auch anhand von XML-Definitionen oder ZemprosysScript-Befehlen übertragen. Somit sind alle komplexen Mechanismen der Softwareinstallationen für Centerix nicht nötig. VPPT-Applikationen laufen über die Client-Software ZCI selbstständig. Die Benutzer des Systems installieren die VPPT-Applikationen nicht und führen nur erlaubte Systemkonfigurationen durch.

• **UMS** (User Management System). Benutzerverwaltung (Erteilung von Rechten, Verwaltung von Passwörtern und Benutzergruppen, Personalisierung, Anwendereinstellungen usw.)

## ZBS (ZEMPROSYS DATABASE SERVER)

ZBS ist ein Serverprogramm für ein komplexes relationales Datenbanksystem. Dieses Datenbanksystem repräsentiert das Backend von Centerix, das in folgenden Datenbanken unterteilt wird:

• **RDB** (Relational Database). Zum Verwalten von unternehmenswichtigen Daten.

• **XDB** (XML Database). Zum Verwalten von unternehmenswichtigen XML-Definitionen.

• **PDB** (Project Database). Zum Verwalten von unternehmenswichtigen Projektdaten.

• **SDB** (Centerix Database). Die Centerix-Datenbank verwaltet jene Daten, die zwar für das Unternehmen wichtig sind, die aber in Verbindung mit der Centerix-Applikationen stehen (siehe SAPP)

• **UDB** (User Database). Zum Verwalten von Benutzerinformationen.

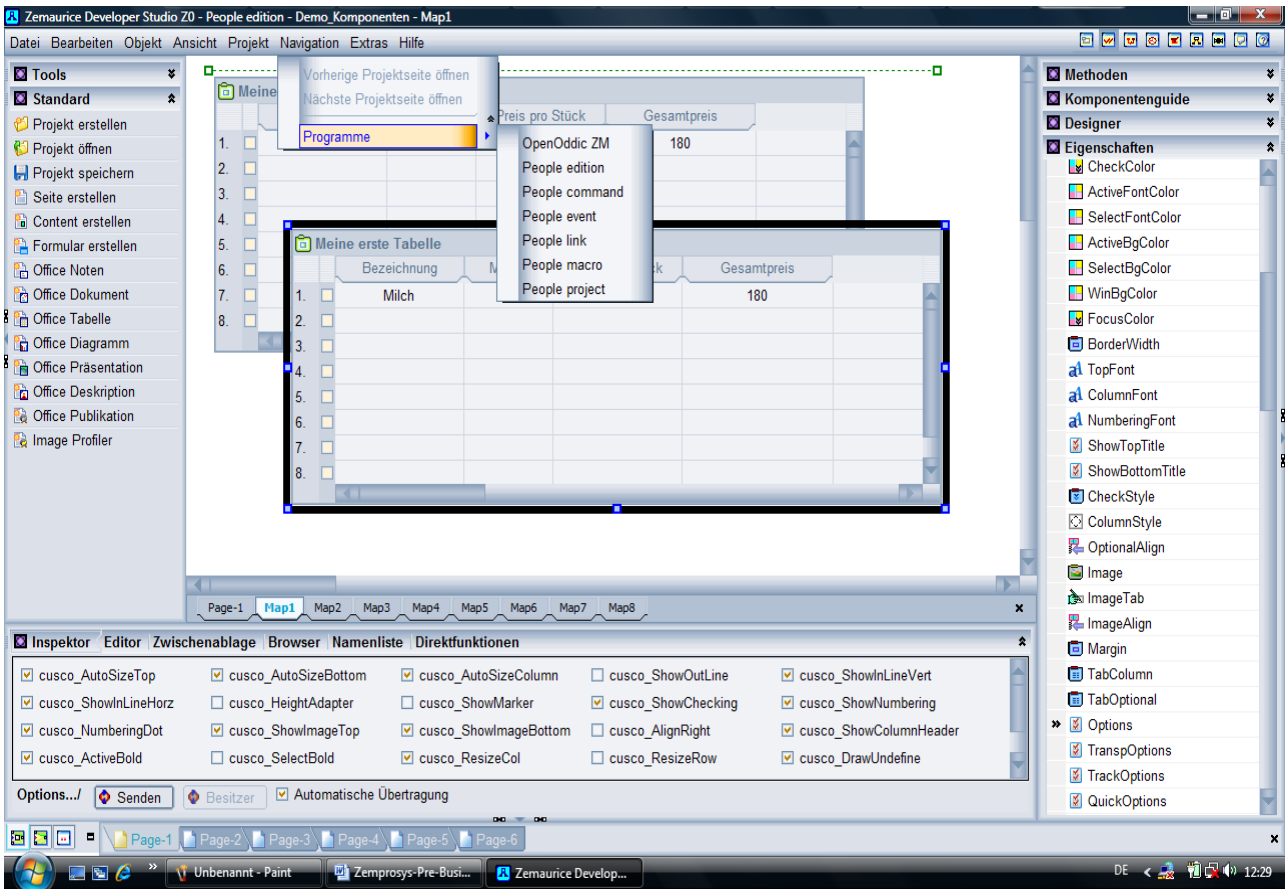
• **EAPP** (Enterprise Applications Database). Definiert die Datenbank für interne Software des Unternehmens und setzt sich nur aus VPPT-Applikationen zusammen. VPPT-Applikationen stehen in XML-Format oder in ZemprosysScript-Form zur Verfügung.

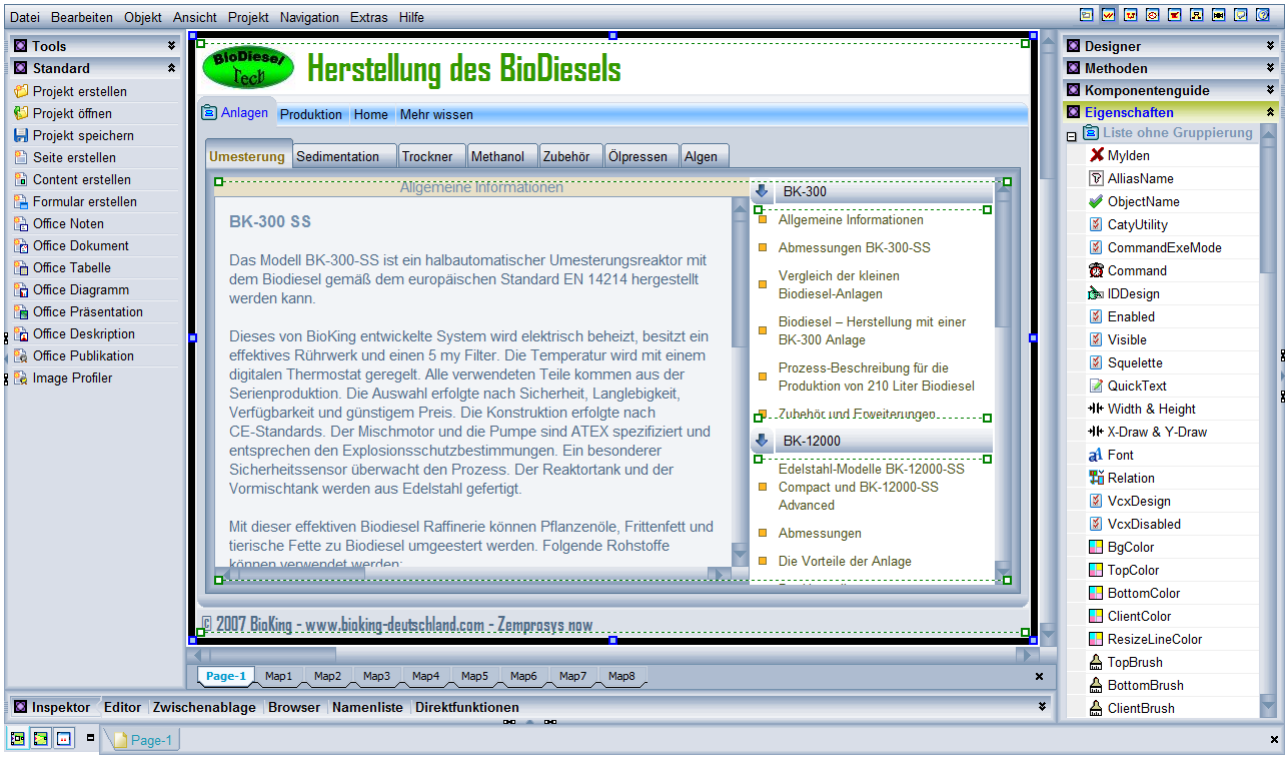
• **SAPP** (Centerix Applications Database). Definiert die Datenbank für VPPT-Applikationen (XML oder ZemprosysScript), die als Centerix-Applikationen betrachten werden. Eine Centerix-Applikation ist eine Free Software. Wenn beispielsweise Centerix von zwei Unternehmen E1 und E2 betrieben bzw. benutzt wird, bekommen die beiden Unternehmen die gleichen Centerix-Applikationen kostenlos, während bei EAPP die Nutzungsrechte an Applikationen mit einer EAPP-Lizenz verknüpft sind.

• **PED** (Project Editors Database). Definiert die Datenbank für VPPT-Applikationen (XML oder Zemprosys Script), die die Abwicklung und Steuerung von Projekten unterstützen. Diese Applikationen stellen die erforderlichen Projektwerkzeuge den Clients bereit. Die Nutzungsrechte an Applikationen ist mit einer PED-Lizenz verknüpft.

• **ARS** (Archive Systems Database). Archivsystem für externe Dateien.

• **REP** (Repository Database). Repository für Komponentenvorlagen und Konfigurationen.


Zemaurice Developer Studio Z0 - People edition - Demo_Komponenten - Map1
Datei Bearbeiten Objekt Ansicht Projekt Navigation Extras Hilfe
Tools
Standard
Projekt erstellen
Projekt öffnen
Projekt speichern
Seite erstellen
Content erstellen
Formular erstellen
Office Noten
Office Dokument
Office Tabelle
Office Diagramm
Office Präsentation
Office Deskription
Office Publikation
Image Profiler
Vorherige Projektseite öffnen
Nächste Projektseite öffnen
Programme
OpenOddic ZM
People edition
People command
People event
People link
People macro
People project
Preis pro Stück
Gesamtpreis
180
Meine erste Tabelle
Bezeichnung
Milch
Methoden
Komponentenguide
Designer
Eigenschaften
CheckColor
ActiveFontColor
SelectFontColor
ActiveBgColor
SelectBgColor
WinBgColor
FocusColor
BorderWidth
TopFont
ColumnFont
NumberingFont
ShowTopTitle
ShowBottomTitle
CheckStyle
ColumnStyle
OptionalAlign
Image
ImageTab
ImageAlign
Margin
TabColumn
TabOptional
Options
TranspOptions
TrackOptions
QuickOptions
Page-1 Map1 Map2 Map3 Map4 Map5 Map6 Map7 Map8
Inspektor Editor Zwischenablage Browser Namenliste Direktfunktionen
cusco_AutoSizeTop
cusco_AutoSizeBottom
cusco_AutoSizeColumn
cusco_ShowOutLine
cusco_ShowInLineVert
cusco_ShowInLineHorz
cusco_HeightAdapter
cusco_ShowMarker
cusco_ShowChecking
cusco_ShowNumbering
cusco_NumberingDot
cusco_ShowImageTop
cusco_ShowImageBottom
cusco_AlignRight
cusco_ShowColumnHeader
cusco_ActiveBold
cusco_SelectBold
cusco_ResizeCol
cusco_ResizeRow
cusco_DrawUndefine
Options.../
Senden
Besitzer
Automatische Übertragung
Page-1 Page-2 Page-3 Page-4 Page-5 Page-6
Unbenannt - Paint
Zemprosys-Pre-Busi...
Zemaurice Develop...
DE
12:29


Datei Bearbeiten Objekt Ansicht Projekt Navigation Extras Hilfe
Tools
Standard
Projekt erstellen
Projekt öffnen
Projekt speichern
Seite erstellen
Content erstellen
Formular erstellen
Office Noten
Office Dokument
Office Tabelle
Office Diagramm
Office Präsentation
Office Deskription
Office Publikation
Image Profiler
BioDiesel Tech
Herstellung des BioDiesels
Anlagen Produktion Home Mehr wissen
Umesterung Sedimentation Trockner Methanol Zubehör Ölpressen Algen
Allgemeine Informationen
BK-300 SS
Das Modell BK-300-SS ist ein halbautomatischer Umesterungsreaktor mit dem Biodiesel gemäß dem europäischen Standard EN 14214 hergestellt werden kann.
Dieses von BioKing entwickelte System wird elektrisch beheizt, besitzt ein effektives Rührwerk und einen 5 my Filter. Die Temperatur wird mit einem digitalen Thermostat geregelt. Alle verwendeten Teile kommen aus der Serienproduktion. Die Auswahl erfolgte nach Sicherheit, Langlebigkeit, Verfügbarkeit und günstigem Preis. Die Konstruktion erfolgte nach CE-Standards. Der Mischmotor und die Pumpe sind ATEX spezifiziert und entsprechen den Explosionsschutzbestimmungen. Ein besonderer Sicherheitssensor überwacht den Prozess. Der Reaktortank und der Vormischtank werden aus Edelstahl gefertigt.
Mit dieser effektiven Biodiesel Raffinerie können Pflanzenöle, Frittenfett und tierische Fette zu Biodiesel umgeestert werden. Folgende Rohstoffe
BK-300
Allgemeine Informationen
Abmessungen BK-300-SS
Vergleich der kleinen Biodiesel-Anlagen
Biodiesel – Herstellung mit einer BK-300 Anlage
Prozess-Beschreibung für die Produktion von 210 Liter Biodiesel
BK-12000
Edelstahl-Modelle BK-12000-SS Compact und BK-12000-SS Advanced
Abmessungen
Die Vorteile der Anlage
© 2007 BioKing - www.bioking-deutschland.com - Zemprosys now
Designer
Methoden
Komponentenguide
Eigenschaften
Liste ohne Gruppierung
Mylden
AliasName
ObjectName
CatyUtility
CommandExeMode
Command
IDDesign
Enabled
Visible
Squelette
QuickText
Width & Height
X-Draw & Y-Draw
Font
Relation
VcxDesign
VcxDisabled
BgColor
TopColor
BottomColor
ClientColor
ResizeLineColor
TopBrush
BottomBrush
ClientBrush
Page-1 Map1 Map2 Map3 Map4 Map5 Map6 Map7 Map8
Inspektor Editor Zwischenablage Browser Namenliste Direktfunktionen
Page-1