S-Square - LowCode/NoCode (LC/NC) Enabling Technology Presentation
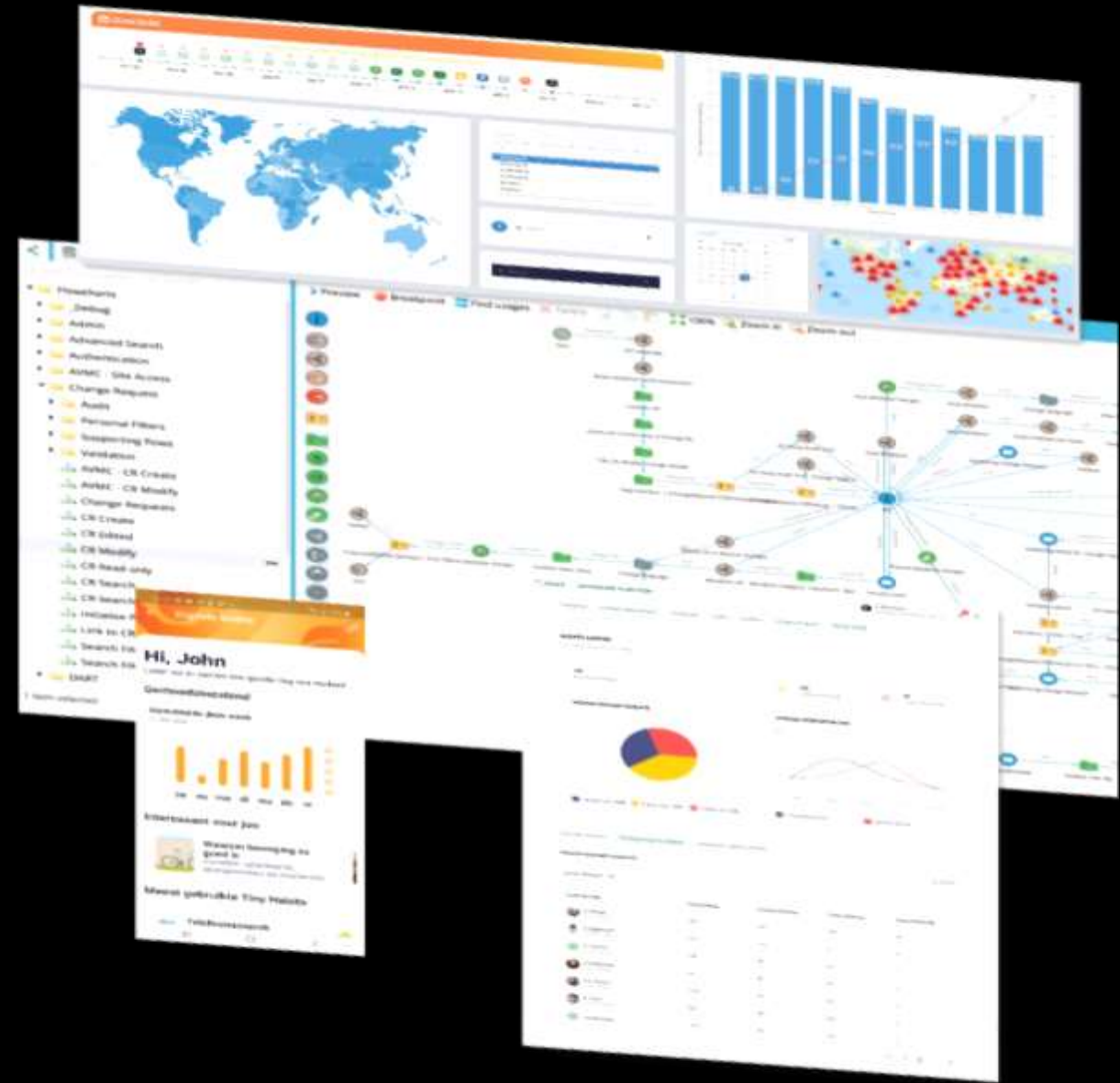
Jeff Friedman,
VP, Sales & Customer Success

Current Challenges in Traditional Application Development

Long Development Timelines

- Custom development with standard SDLC processes

- Long incubation period before seeing a MVP

- Minor changes require long turn around time for design, build and testing.

High Capital Expenditure and Operating Costs

- Investment in Software platforms and Infrastructure for custom development

- Higher support costs due to diverse support requirements

Disparate Technology Landscape

- Multiple small projects using disparate technologies

- No uniform platform to manage small developments

Developer Shortages

- Developer shortages and skill-set challenges

- Multiple small productivity projects get deprioritized

6 Generations of Programming Languages

First generation (1GL) - machine-level programming language used to program first-generation computers
Examples: machine-level programming languages

Second generation (2GL) - assembly languages.   Examples: Assembly

Third generation (3GL) - more machine-independent (portable) and more abstract therefore more programmer-friendly than previous generations of languages
Examples: Fortran, COBOL, BASIC, Pascal, C, C++, Perl, Python, Java, JavaScript, Ruby, PHP, C#

Fourth generation (4GL) - include support for database management, report generation, mathematical optimization, GUI development, or web development. Examples: ABAP, Unix Shell, SQL, PL/SQL, Oracle Reports, R

Fifth generation (5GL) - any programming language based on problem-solving using constraints given to the program to make the computer solve a given problem without the programmer, rather than using an algorithm written by a programmer.   Examples: Prolog, OPS5, Mercury

Sixth generation (6GL) - programming language based on visual development. The overall umbrella term for these is "NoCode".   Examples: Appian, WEM.io, Bubble.io
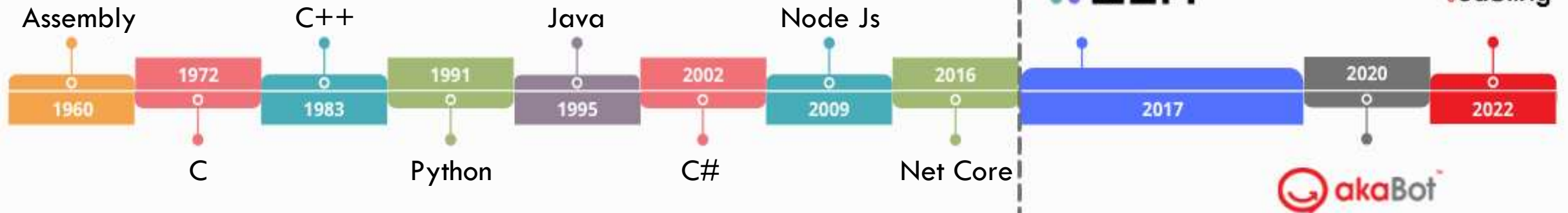
Reinventing Software Development

Traditional Coding
Requiring expensive, hard to retain code-linquists

No-Code
Empowering transforming support to employ business-knowledgeable techno-functional resources

Assembly   C++   Java   Node Js

WEM   edSling

| 1972 | 1991 | 2002 | 2016 | 2020 |
| 1960 | 1983 | 1995 | 2009 | 2017 | 2022 |

C   Python   C#   Net Core

akaBot

Traditional computer languages require programmers to translate their thinking process into code built for the CPU and memory

Optimized for how we humans think. Converting natural thinking process into working software

Digital Transformation.
Legacy Modernization.
Business Velocity.

80%
COST REDUCTION

Empowers employing business knowledgeable (techno-functional) resources instead of costly, hard to retain code-linquists to build, deploy and maintain secure scalable enterprise-grade software.

10%
FASTER TIME-TO-MARKET

View app development in real-time. Deploy and update applications with a single click. Deliver software 10 times faster than traditional programming methods.
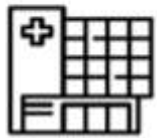
100%
ALIGNED TO BUSINESS

Translate innovative business ideas to custom software built with no code app builder at the speed of, and fully aligned with, business requirements.

Banks, Financial Services and Insurance >

Healthcare >

Telecommunication >

Education & Training >
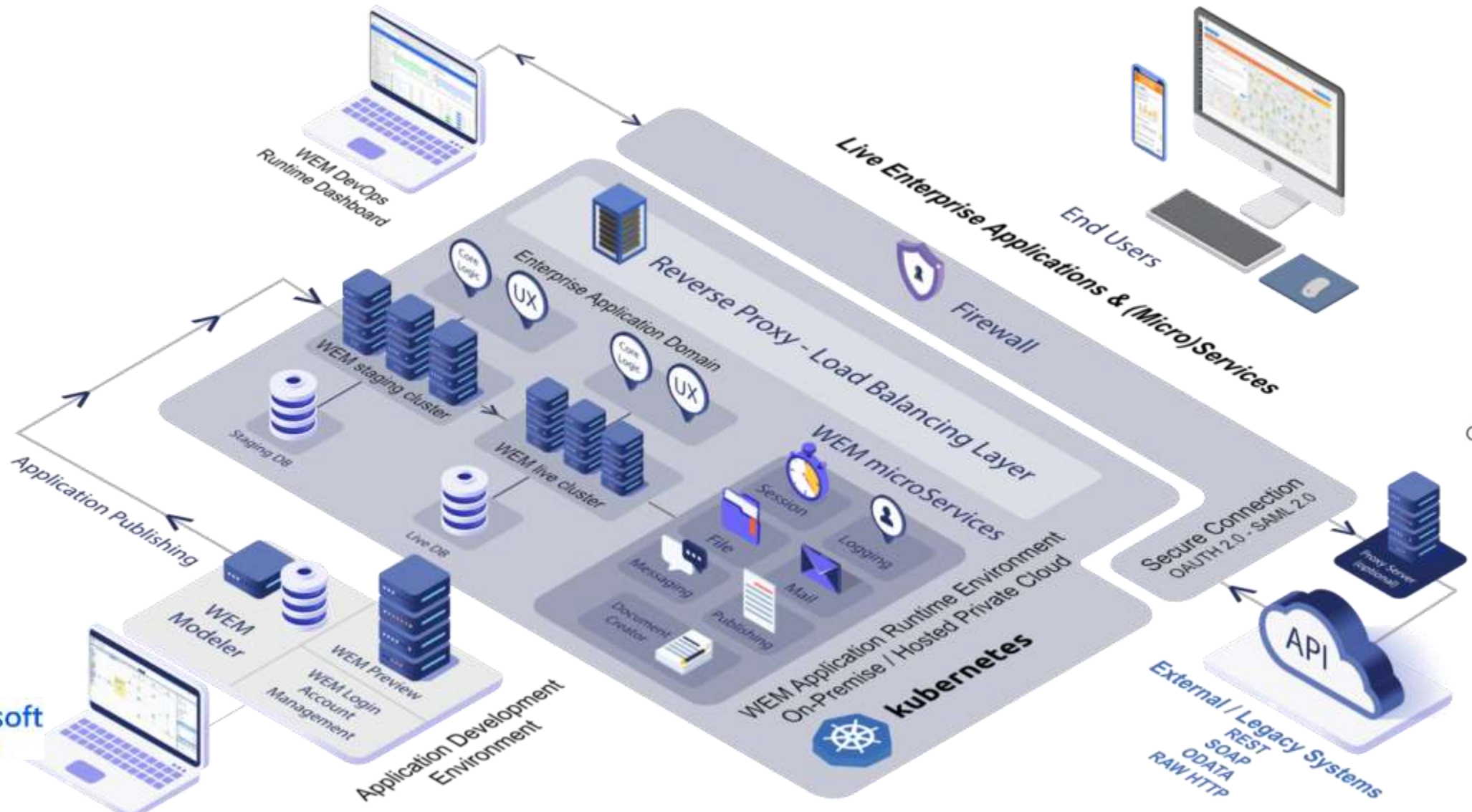
Manufacturing >

Public Sector >

Automotive >

Real Estate >

SCALABLE, SECURE CLOUD ARCHITECTURE

FLEXIBLE DEPLOYMENT OPTIONS FOR SHARED HOSTING, ON-PREMISE APPLIANCE AND PRIVATE APPLICATION CLOUD

Criteria	WEM	Betty Blocks	Power Apps	OutSystems	Mendix
Category	No Code	Low code	Low code	Medium to high code	Low code
Platforms	Web, native apps	Web apps	Web, native apps	Web, native apps	Web, native apps
Data Model	Drag & Drop	Visual Editor	Tables	Visual Editor	Visual editor
Visual Editor	Web-based	For backend apps	Web-based	Many designer	Web-based, desktop-based
Workflows	Drag & Drop	Action Modeler	MS Flow	Visual modeler	Visual modeler
Look & Feel	Custom templates	Custom js/css/html	Customizable	Custom js/css	Custom js/css
Environment	Public, private cloud, on premise	Public cloud, on premise	Public, private cloud, on premise	Public, private cloud, on premise	Public, private cloud, on premise
Release Management	Fully	Fully	Partially	Fully	Fully
Integration	All API standards	JSON, SOAP/REST	Office365, REST	SOAP/REST	SOAP/REST

Use Case – Voice Enabled Digital Transformation and Legacy System Integration

The VGZ Group, headquartered in the Netherlands carries out health insurance for their different brands, such as Univé, IZZ, and Bewuzt. They employ more than 2700 people, with total revenue of € 10,8 billion.

CUSTOMER CHALLENGES

- Since 2005, the company used different legacy systems on various locations for their different brands. The challenge was to maintain the source code and content of these legacy systems which were built with traditional coding.
- There was a limited timeframe, it had to be live within six months due to regulatory pressure.
- Centralization of all the information.

WEM ADVANTAGES

- **Agile development, week to week results, short time to market (live in 6 months)**
- **From a complex and difficult to maintain legacy system to a clear, easy to use and no-code environment to be maintained by non-IT skilled employees**
- **Use of existing data from legacy systems**
- **Integration with Lucene search engine library**
- **This is a cloud solution that offers flexible workspaces (not tied to a location)**
- **Easy to extend the application**

PROBLEM

The company needed a powerful and easy-to-use search engine for mining into a large amount of insurance data. There was a corporate need to centralize the customer service systems for all brands so that all the brands had one clear overview of all the necessary information. Since 2005, the company used different legacy systems on various locations for their different brands. These legacy systems were built using traditional programming languages (using code).

SOLUTION

WEM integrated the project with Lucene for a powerful cross-platform full-text search. This was a highly complex integration project. The end-user experienced it as a very easy way of searching. All the data was linked to an extensive authorization mechanism so that only authorized persons can access the information. This was a highly complex structure that was used throughout the application. The use of a complex ontology setup to support the knowledge management features added a layer of complexity.

Representative WEM Enterprise Customers

Thank You

Jeff Friedman,
VP, Sales & Customer Success

S-Square Systems, Inc.

4225 Executive Square Suite 600
La Jolla, CA 92037
+1 858-213-7063, +1 858-764-4441

S-Square

TRUSTED . TESTED . COMMITTED