S-Square - LowCode/NoCode (LC/NC) Enabling Technology Presentation

Jeff Friedman,
VP, Sales & Customer Success

Version - 20221215_V1

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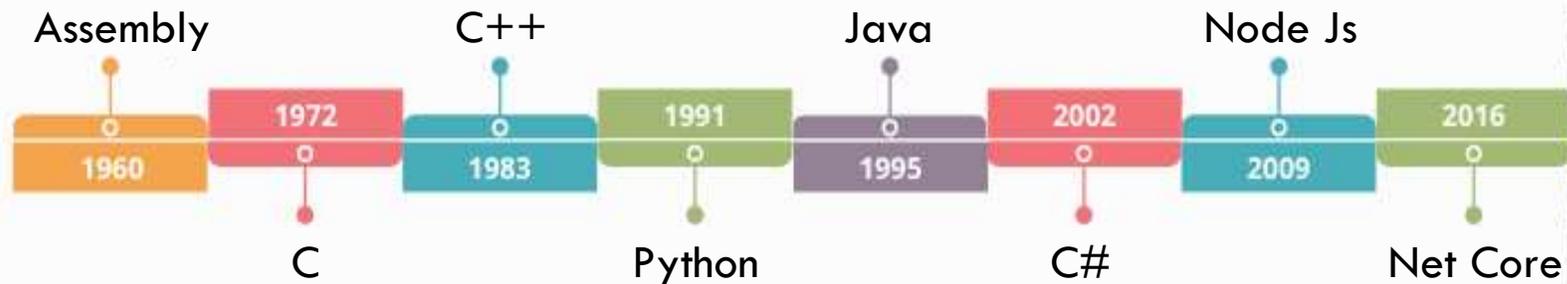
Fifth generation (5GL) - any programming language based on problem-solving using constraints given to the program to make the computer solve a given problem without the programmer, rather than using an algorithm written by a programmer.   Examples: Prolog, OPS5, Mercury

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

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

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

C

Python

C#

Net Core

WEM

edSling

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.
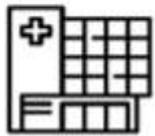
100%
ALIGNED TO BUSINESS

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 >
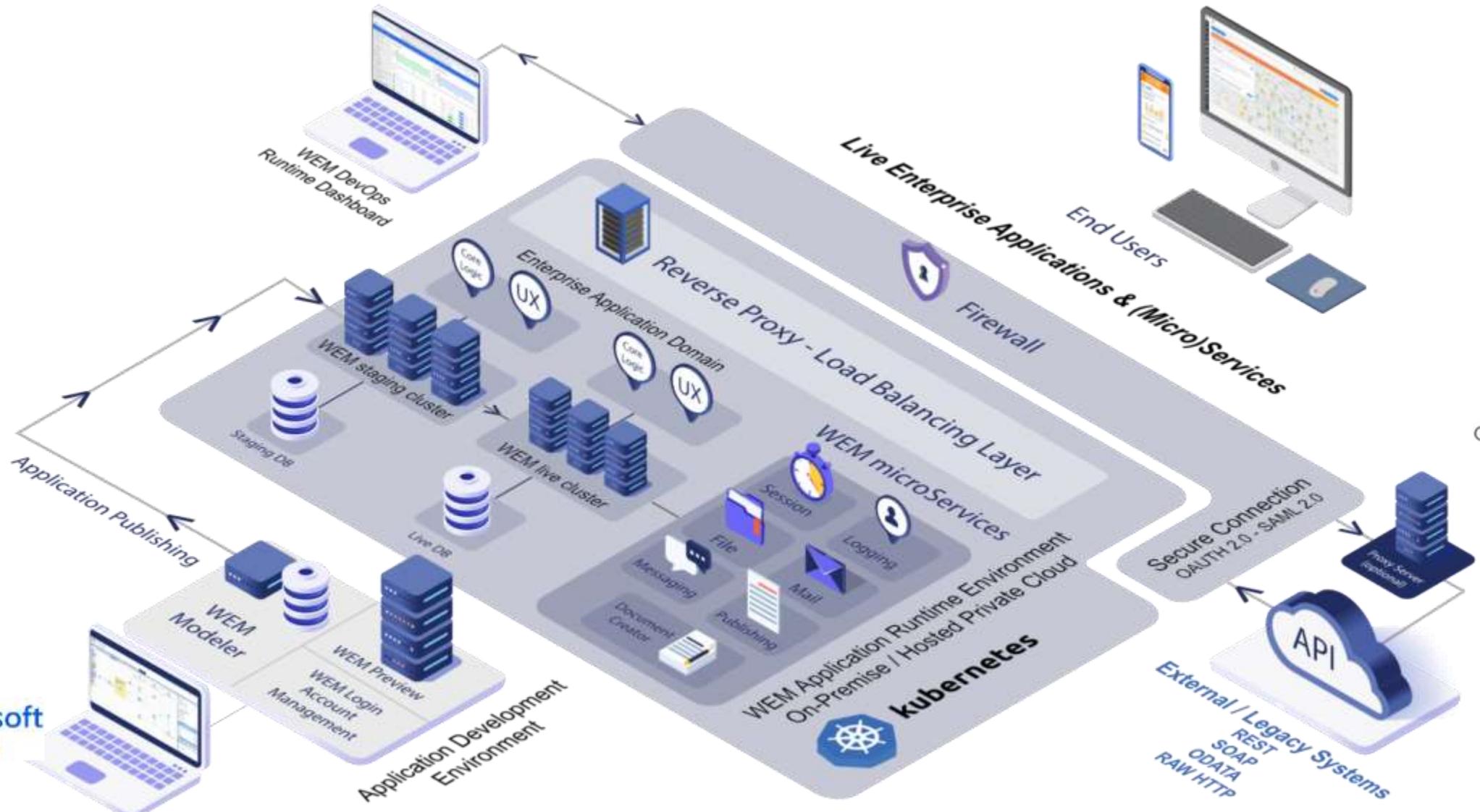
Automotive >

Real Estate >

SCALABLE, SECURE CLOUD ARCHITECTURE

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

3rd Party LCNC Marketplace Product Evaluation

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 – HR On-Off Boarding Process Automation utilizing RPA

This is an international Engineering & Manufacturing company with more than 5000 employees, headquartered in Georgia, USA.

PROBLEM

CUSTOMER CHALLENGES

- A high manual effort was required to perform the on-offboarding process.
- These processes had a long processing time and were prone to errors.
- There were several repetitive and time-consuming tasks involved in collecting and consolidating data from ServiceNow tickets, and syncing data on multiple systems such as ServiceNow, Active Directory and Office 365.

The company realized that a high manual effort was being put in to perform the onoffboarding processes. Due to the lengthy and manual nature of the onboarding process, the new joiners were not able to start work immediately after joining. Similarly, the administration was often not able to complete the offboarding process in time.

SOLUTION

WEM ADVANTAGES

- **RPA deployment reduced the effort by 90%**
- **Processing time was reduced by 85%**
- **Errors were reduced by 100% with RPA robots.**
- **RPA Robot is available 24/7**
- **Reduce operational cost and improve productivity by 95%**

We deployed RPA Robots (akaBot) to automate the Onboarding and Offboarding process. The deployed RPA Robots can emulate human actions on a computer to perform the OnOffboarding process. All the mundane and repetitive actions within the process were automated by RPA Robots. The average process completion time was reduced to 5 minutes from 60 minutes.

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