Vanilla Corp. – Preparing to Release Solution into Production

Recommendations & Next Steps

Celia Muriel – 9th October 2019

Solutions Architect

celiamuriel.com

Goal Of This Presentation

- Vanilla Corp. will soon release an major modification in the architecture of their IT ecosystem to better support their core business
- Role of Vanilla Corp.’s IT solution within Vanilla Corp.’s ecosystem
- Status of Vanilla Corp.’s Project
- Deadlines
- Discuss on how to release our solution into production and maintain it in order to most leverage and benefit out of IT investments and support accomplishment of business objectives

Agenda

- Ecosystem Overview
- Vanilla Corp. Ecosystem Monitoring
- Status Of Our IT Solution
- Expected Used Of Our IT Solution
- Performance Testing
- Governance & Security
- Objectives, Recommendations & Next Steps

Vanilla Corp. Ecosystem Architecture

Overview of Vanilla Corp.'s Ecosystem Architecture.

20191009_RFID sensor within IT infrastructure.

It is a large architecture, with many systems involved, and interconnected. In order for the architecture to work swiftly, every system must provide with response in a timely manner.

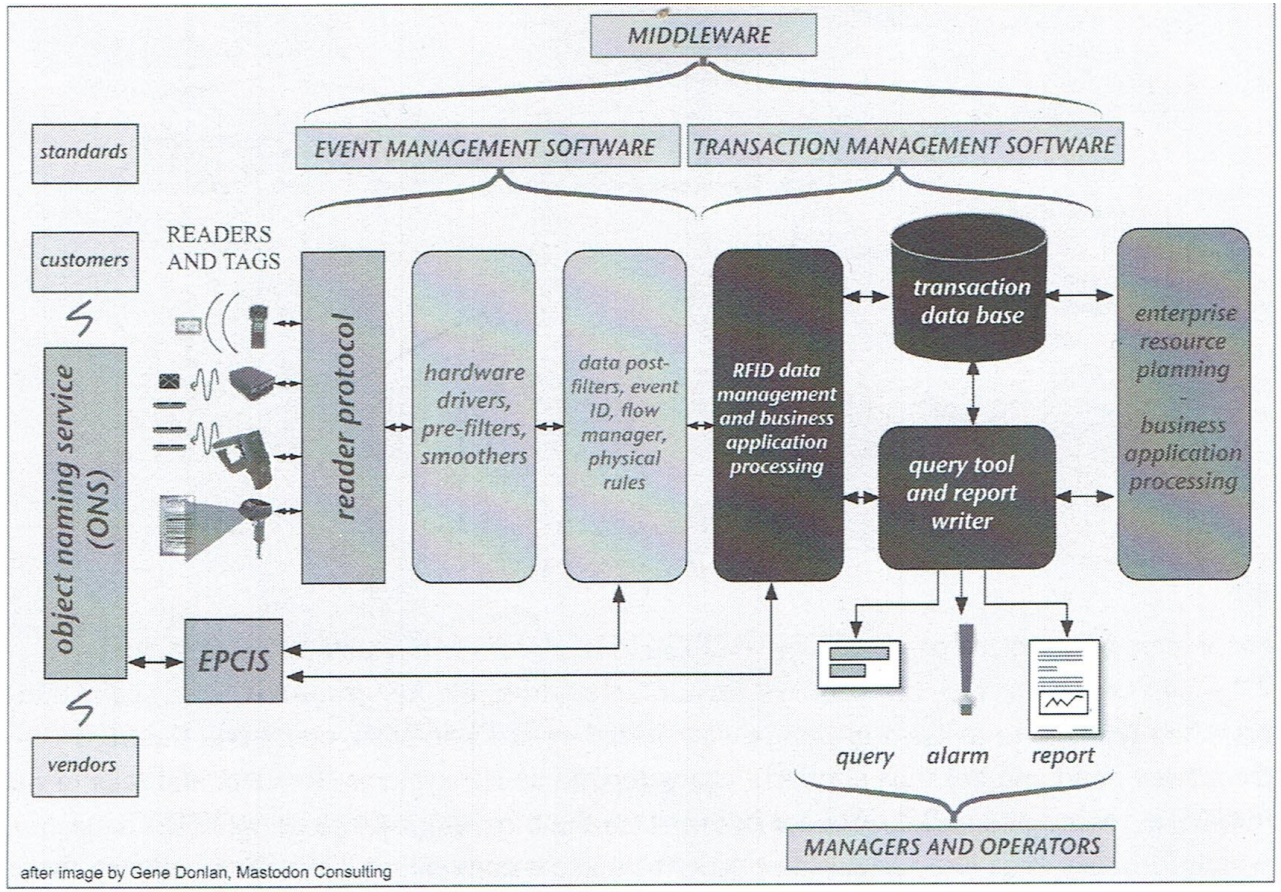


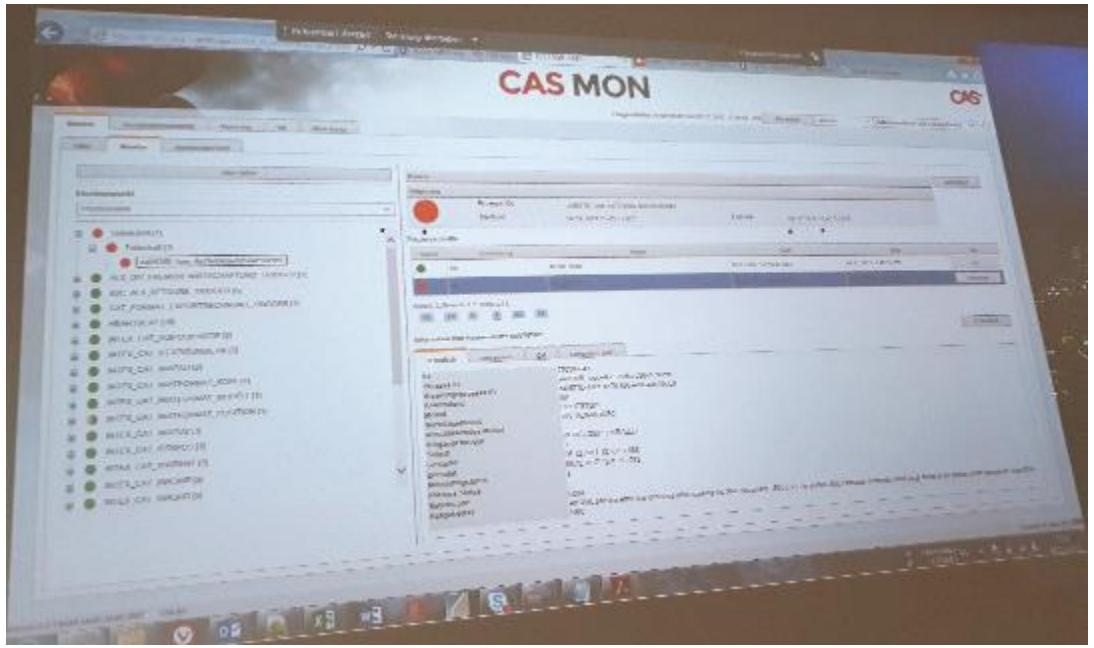
Image via "The RF in RFID. UHF RFID in Practice" by Daniel M. Dobkin. Newnes, Elsevier. Second Edition

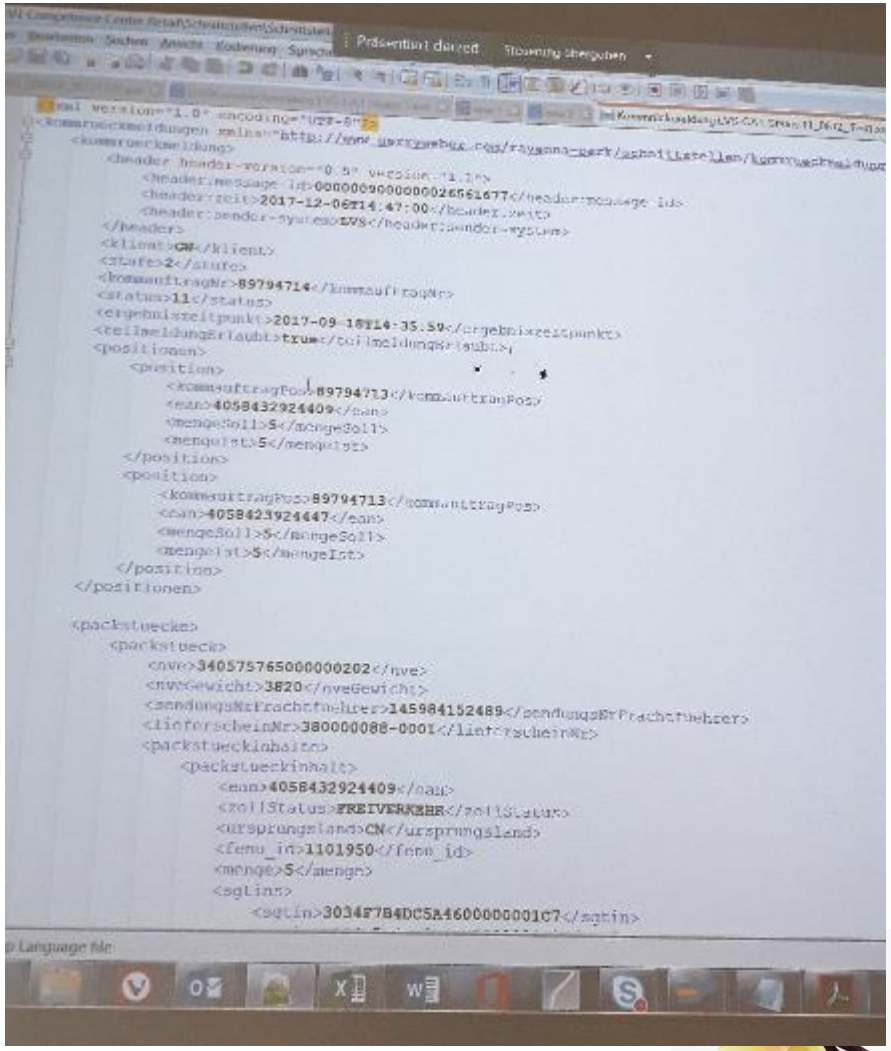
Mission Critical Environment*

* **Essential** to the **survival** of Vanilla Corp.'s **core business**

Vanilla Corp. Ecosystem Monitoring

Vanilla Corp. has deployed a monitoring tool to ensure availability and overall performance of processes running in the ecosystem
It does not diagnose specific uses which cause delays or errors → the **stakeholder** of each **component** must **provide** with a **timely reply and solution** for any incident





Proposed Tests Prior to Releasing into Production

- **Functional testing**.
  - Components have been tested in a lab environment with low data and tag volumes, but it is necessary to test real data and high-volumes
- **Performance testing** to ensure that the components perform adequately.
- **Load testing** to ensure that the components perform under anticipated load.
- **Endurance testing** to ensure that the components perform under load over an extended period of time.
- **Disaster recovery** to ensure that the system can be recovered in the event of some kind of disaster.
- The above applies to both **server** and **client side** components

Goal Of Preparation Efforts To Release Into Production I

- Most leverage and benefit out of our solution and **support accomplishment of business objectives**
- Maintain **data quality**
- Estimate **how long initial loads** and **daily loads** are going to take
  - Use for planning purposes during rollout
- Estimate initial **infrastructure capacity** requirements
- Respond to the question **what to do** when the amount of **data** or **concurrency** in the solution **increases**
- **Scalability**
- **Workaround to lose network connection** in stores
- Provide with a **timely reply** and solution when **issues** arise

Goal Of Preparation Efforts To Release Into Production II

- **Set expectations about performance and capacity** as test results will be obtained in a particular infrastructure, which may be different on the Production system
  - Factors which influence results
    - Middleware version and configuration
    - Physical systems/ virtual machines
    - If virtual machines: hypervisor, its configuration, and the hardware underneath it
    - Concurrency on the solution: Reporting, Back Office, Cycle Counts, data loaded from Enterprise Systems and Business Applications
    - Amount of data
    - Operating System and Database version
    - Network

Suggested Performance Tests I

- Make **load tests** to estimate ingestion rates and times to anticipate reply to customer on deployment and set expectations
  - REST API and Web Services
  - Make several tests where the amount of data is larger in every test
- **Concurrency tests** to anticipate to customer's concerns and set expectations
  - RFID/ Mobile devices and Middleware
  - More than one call in the REST API
  - Web Services
  - All the above at the same time
- **Disaster Recovery** scenario
- **Full capacity tests** (space, CPU, I/O) to understand the solution's behaviour during peak timeframes, high season or unexpected events
  - Run heavy workloads in all components simultaneously
- **Lose network connection** at stores

Suggested Performance Tests II

- **Duplicate** and **null IDs**
  - Reference number
  - deliveryNumber
  - Nummer der Verpackungseinheit (NVE) or Serial Shipping Container Code (SSCC)
  - trackingNumber
  - Electronic Product Code (EPC)
  - European Article Number (EAN)
  - Advanced Ship Notice (ASN)
- **Foreign keys which don’t exist in the master table**
- **Wrong data formats** and **wrong data types**
  - IDs
    - Length, particular formats as the barcodes, etc.
  - Dates, timestamps and times
    - No UTC, no ‘YYYY-MM-DD’
  - Expect a number and receive characters, dates, etc.

Governance I

- Disaster Recovery
  - Backups, restores
- Provide with timely reply and solution for incidents observed in the Vanilla Corp. monitoring tool
- Software upgrades and migrations
  - Middleware
  - APIs
  - Back Office
  - Mobile
  - Operating System and Databases
- What to do when the amount of data/ concurrency increases

Governance II

- Procedure to report incidents
  - Customer → Vanilla Corp.
  - Internally within Vanilla Corp.
  - Incident categorization
- Regular maintenance tasks
- Tools in place for troubleshooting
- Data Quality rules
  - Duplicated keys
  - Null keys
  - Foreign keys which don’t exist in the master table
  - Wrong data formats and wrong data types

Security

- Create new users and roles
- Decommission old users and roles
- Passwords
  - Define procedures to change them
  - Format, length
  - Password vault
  - Authorization to have it
- Define procedures and how to grant access to the solution for maintenance tasks or troubleshooting

Immediate Objectives

- Agree the **scope of tests** to be performed (Performance, Load, Endurance etc.)
- Agree **when in the project lifecycle** the various testing activities should be performed.
- Determine who is **accountable and responsible** for the delivery of the testing activities.
- Understand the **likely work effort** required for such testing activities (order of magnitude 10/100/1000 days).
- Agree on follow up activities and meeting cadence.

Slide courtesy of Iain Ritchie

Recommendations and Next Steps

- **Discuss and agree on immediate objectives**
  - Include **task prioritization**
- Involve **Software Engineering/QA teams** in the discussion
  - Measure performance
- Make an **Operations Manual**
  - Details on how to operate Governance and Security
- **Discuss with the customer**
  - Data volumes are expected to be loaded initially and daily to estimate solution requirements as well as the best way to perform certain activities (e.g., initial load, full cycle count)
  - SLAs. E.g., if a system crashes and needs to be restored, how much time it should take? Would 2 days be OK? Is this time SLA feasible with current capabilities, features, support contract and hardware?
  - Identify test environment details where testing is going to be carried on

Recommendations and Next Steps

- Define **test metrics** which are captured and shared
  - Number of test cases
  - Number of defects
  - Number of tests executed against the ones planned
  - Execution time as per test conditions
- **Set expectations**

Comments, please