Computerized Maintenance Management System

J. Barr, Holger Drass, Matthew Rumore for the CMMS team
Special contribution by Alberto Pittolo from Tecnoteca

Agenda

- Introduction (J. Barr)
- Selection process (H. Drass)
- Demo Preventive and Predictive maintenance (A. Pittolo and M. Rumore)
  - Jira connection
  - BIM model (CAD)
  - Warehouse connection for a consumable
- Data Model (H.Drass)
- Ingest Excel Sheet (M. Rumore)
- Backup material

Introduction

–

J. Barr

Introduction - Who are we & What are we doing?

The Company

The Software

In 2022 Rubin selected the Tecnoteca proposal for openMAINT as our **Computerized Maintenance management System**

We have been working with Tecnoteca for about a year to implement openMAINT, an open-source CMMS software to maintain all of the critical Rubin Observatory assets.

We have made significant progress, but there are significant challenges. Observatories, domes, telescopes, cameras, mirrors, coating plant… are not typical applications of facilities maintenance systems.

The Working Group +

- Alberto Pittolo (TecnoTeca PM)
- Holger Drass
- Jeff Barr
- Matthew Rumore
- David Cabrera
- Glenaver Charles Emerson
- Eduardo Serrano
- Bill Schoening
- Diane Hascall
- Andy Clements
- Chuck Claver
- Mario Rivera
- Iain Goodenow
- …and others.
- YOUR help is welcome & needed!

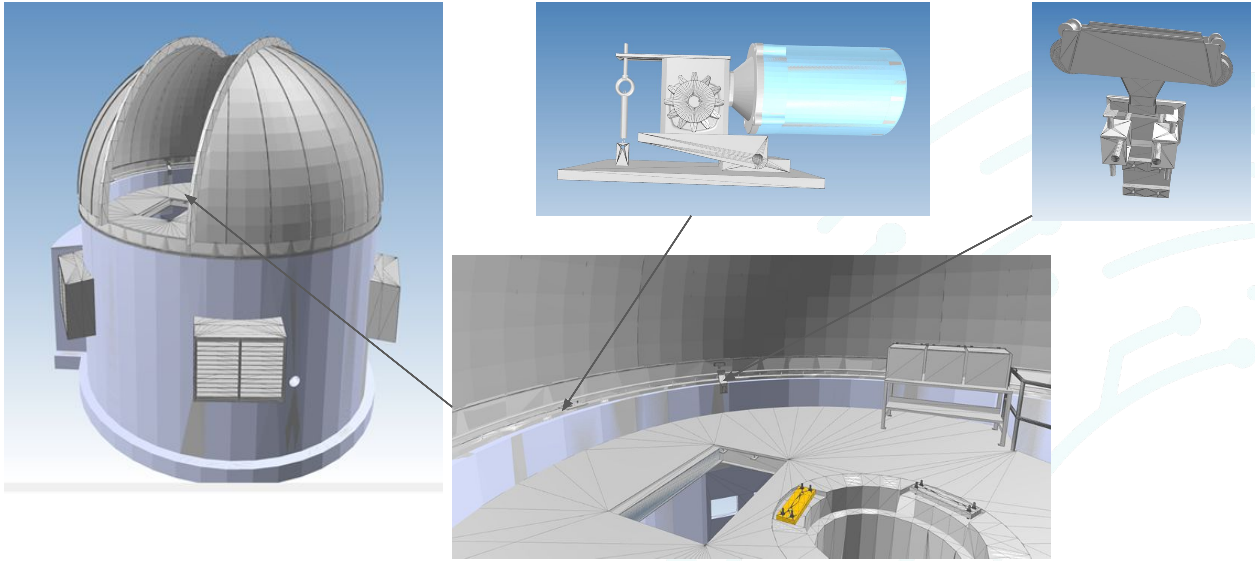
Parts of the CMMS Implementation Process

- **IT Connectivity**
  - Users, Administrators, Jira interface, links to Docushare
- **Decomposition (breakdown) of the Rubin Observatory for maintenance purposes**
  - Maintenance requires a different focus than WBS, model assembly/part trees or other databases
- **Documentation**
  - Rubin and vendors has developed a lot of documentation, but it is not all relevant nor easily referenced
- **3D Modeling**
  - File conversion from SolidWorks and other Rubin standards for OpenMAINT is challenging
- **Ingestion of maintained asset/component definition into openMAINT**
  - Tecnoteca methods and consultation have helped with this
- **Technician interface and testing**
  - Just getting started with field application
- **Yet to come:**
  - Use of openMAINT to help with spares, bench stock, warehouse management
  - OpenMAINT as an aid to our Engineering Facilities Database in troubleshooting and predictive maintenance
  - Training on openMAINT - coming soon!

Auxiliary Telescope as a Starting Point

Within the Rubin ecosystem, Auxtel has instances of most of the types of equipment we need to maintain for the full Rubin Observatory, and is more manageable as a trial run

- Telescope
- Drives
- Mirrors
- Instrument
- Compressors
- Dome
- Shutters
- Building
- Louvers/fans
- Electrical systems



SolidWorks models converted to IFC files for interactive use in openMAINT

Some CMMS Points of Interest

Much of what we are building has been in maintenance for years already:
Auxtel, coating plant, building, HVAC system, air compressors, electrical system…

NOIRLab has now also selected openMAINT as their CMMS and has begun working with Tecnoteca to implement it.

Please begin to think about how this can best be applied for the systems and components you are helping to build when they become delivered assets of Rubin Observatory.

We need not just a Working Group, but a legion of developers as subject matter experts to make this work.

The true eventual metrics of or success will be time saved in maintenance work and mean time between failure of our critical equipment.

CMMS selection process

Computerized Maintenance Management System - CMMS

Motivation

Working group

CMMS Requirements

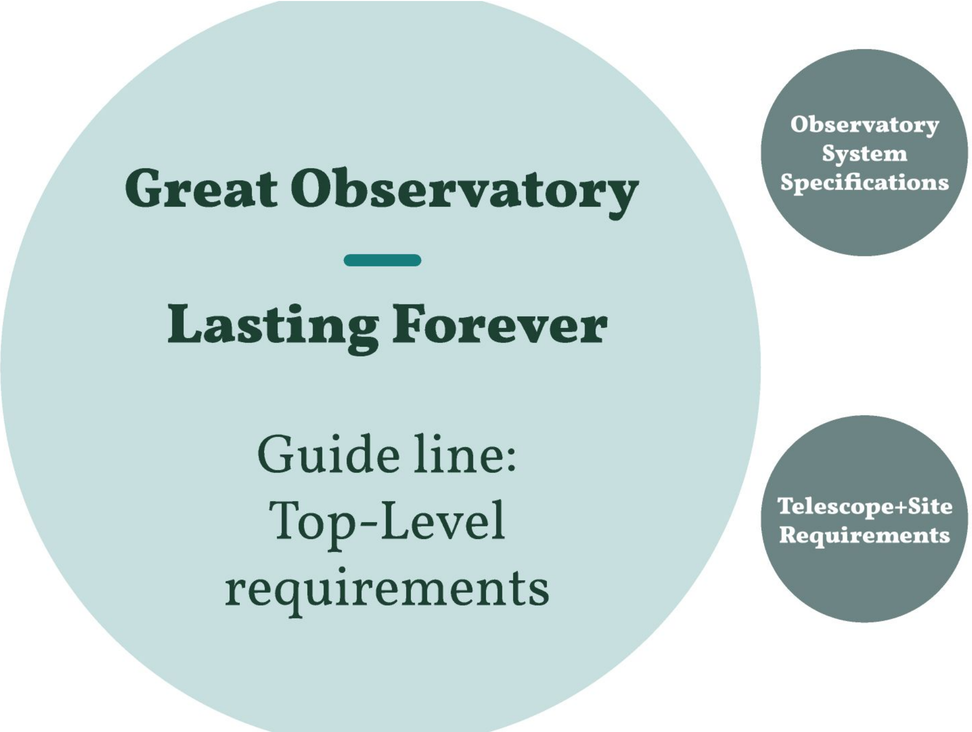
Trade-off analysis

Use cases + Early Design Decisions

Candidate selection

Decision + Next Steps

Motivation



Great Observatory

Lasting Forever

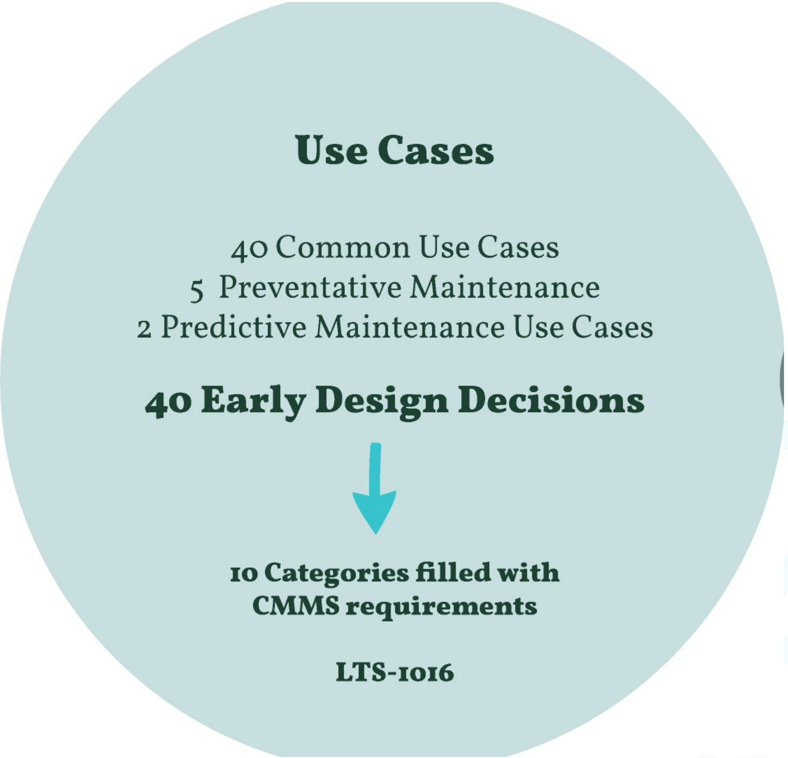
Guide line:
Top-Level
requirements

Observatory System Specifications

Telescope+Site Requirements

Working group

Management = Sandrine Thomas, Jeff Barr, Chuck Claver, Eduardo Serrano
IT = Cristian Silva
Camera = Diane Hascall
Software = Wouter van Reeven
Documentation = Matthew Rumore
Systems Engineering = Austin Roberts, Holger Drass



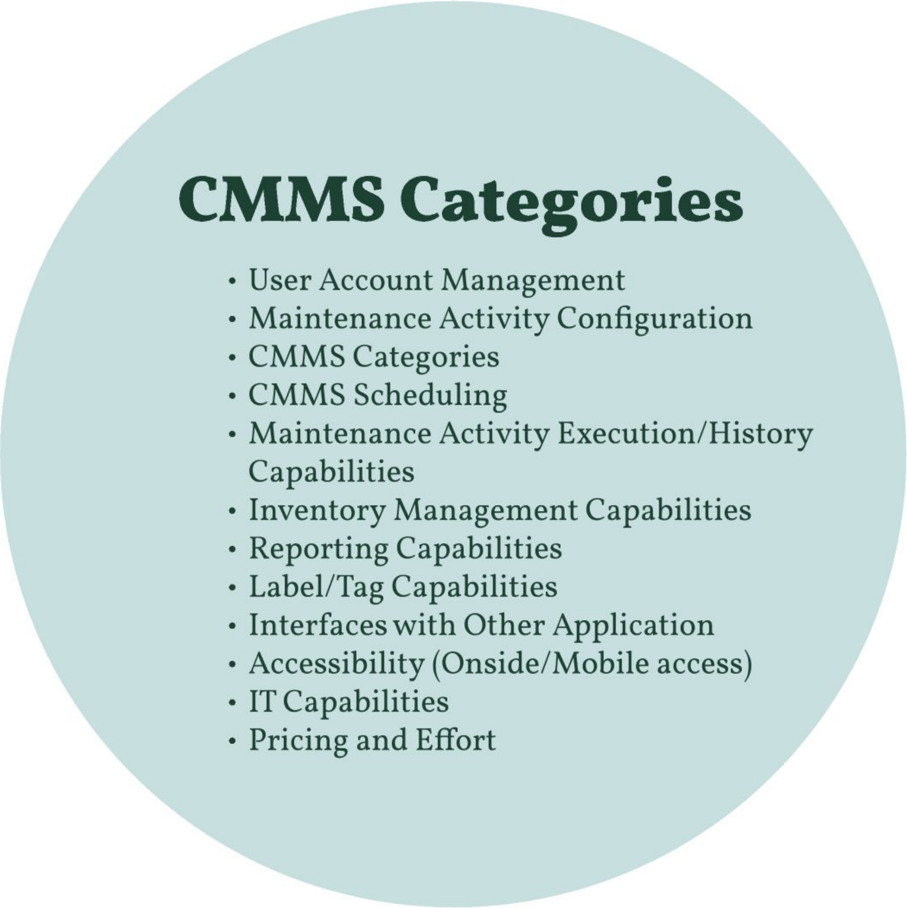
Use Cases

40 Common Use Cases
5 Preventative Maintenance
2 Predictive Maintenance Use Cases

40 Early Design Decisions

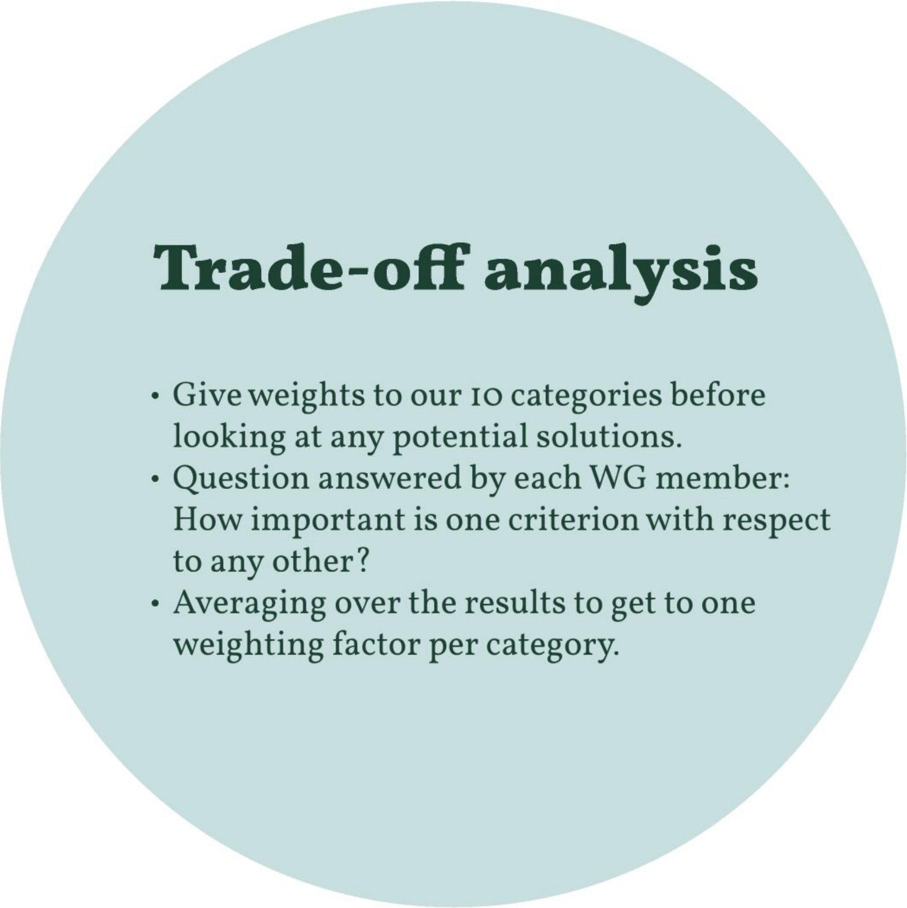
10 Categories filled with CMMS requirements

LTS-1016



CMMS Categories

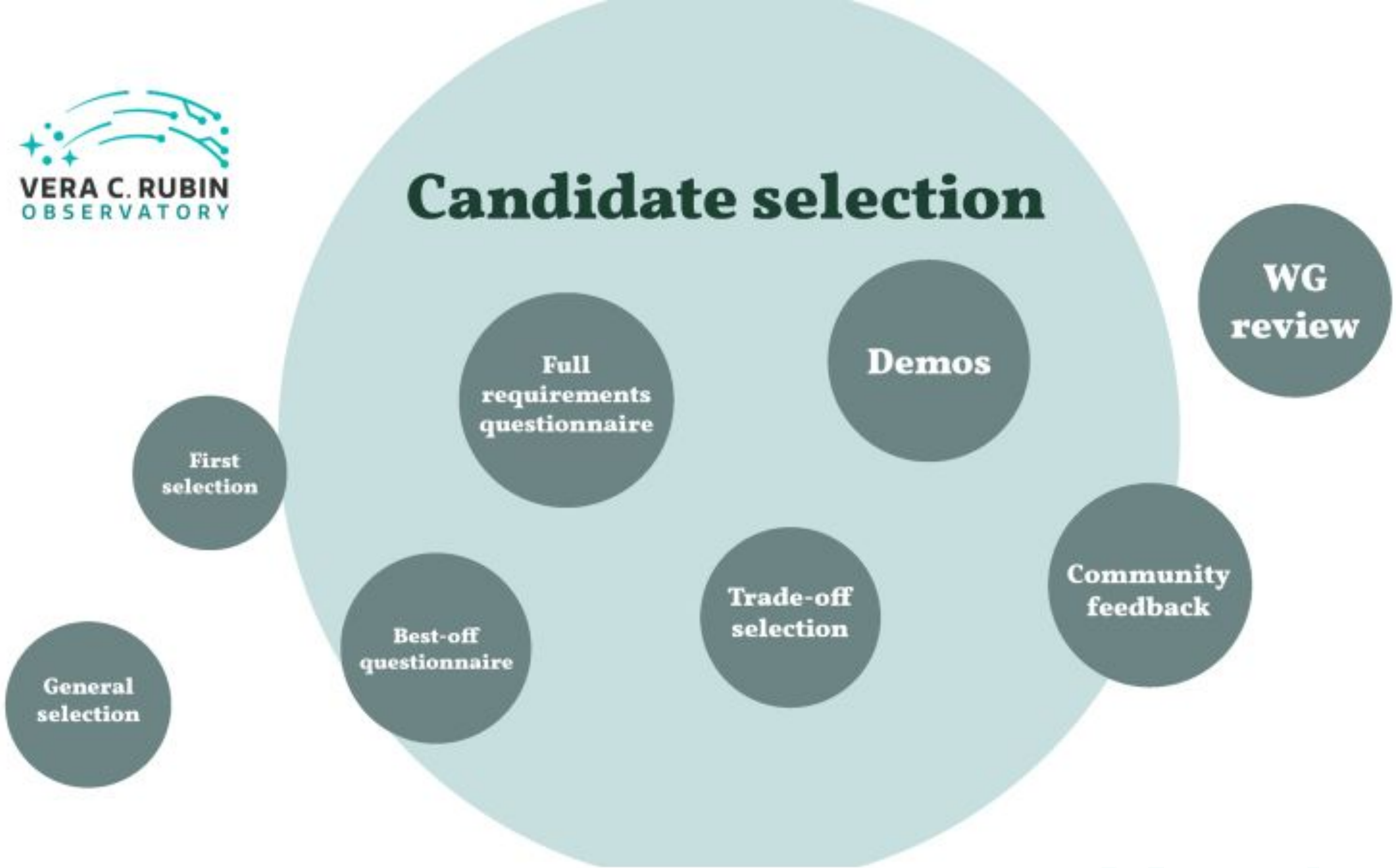
- User Account Management
- Maintenance Activity Configuration
- CMMS Categories
- CMMS Scheduling
- Maintenance Activity Execution/History Capabilities
- Inventory Management Capabilities
- Reporting Capabilities
- Label/Tag Capabilities
- Interfaces with Other Application
- Accessibility (Onside/Mobile access)
- IT Capabilities
- Pricing and Effort



Trade-off analysis

- Give weights to our 10 categories before looking at any potential solutions.
- Question answered by each WG member: How important is one criterion with respect to any other?
- Averaging over the results to get to one weighting factor per category.

Candidate selection



General selection

First selection

Best-off questionnaire

Full requirements questionnaire

Trade-off selection

Demos

Community feedback

WG review

Decision

Tecnoteca with openMAINT.

WG Re-Orga:
Eduardo Serrano + David Cabrera
Summit Specialists for Mechanical, Electrical and Electronics Engineering

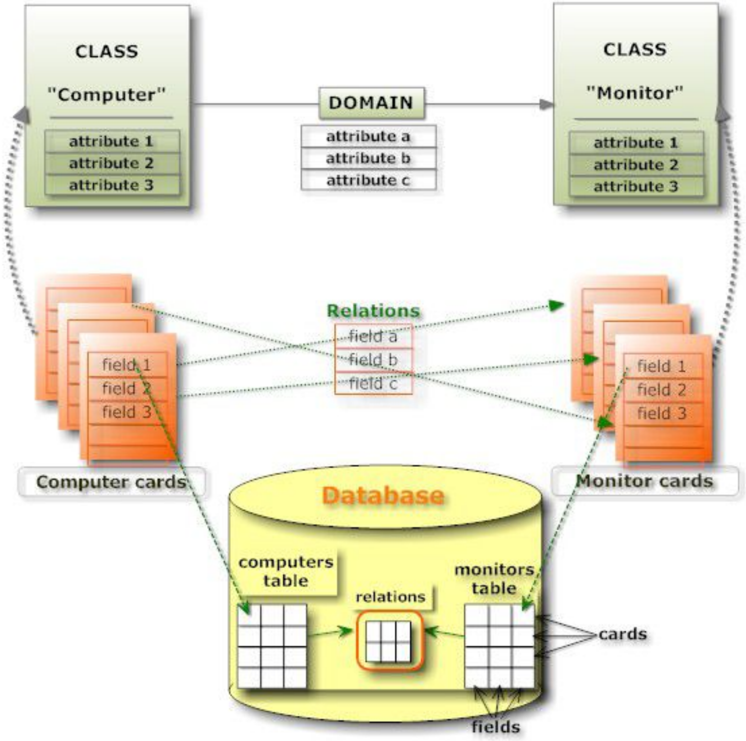
- ✓ Installation
- □ Configuration + Testing
- □ Requirements Verification

Thanks IT-Team!

Demonstration
Preventive / Predictive
Maintenance
–
Albert Pittolo and M. Rumore

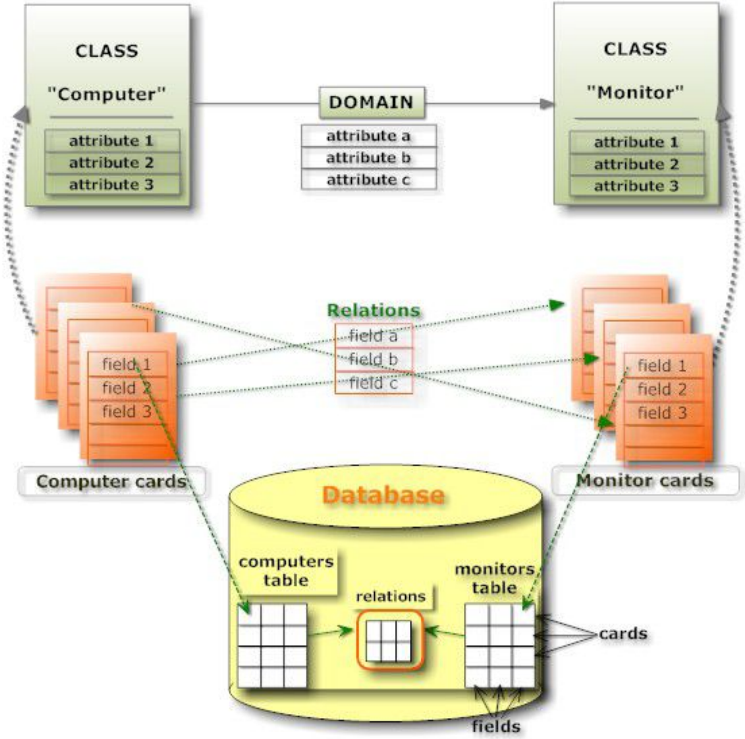
CMMS
Data Model
–
H.Drass

Data Model



- CMMS uses Object-Relational Database
- Needs **entity-relationship model**

Data model



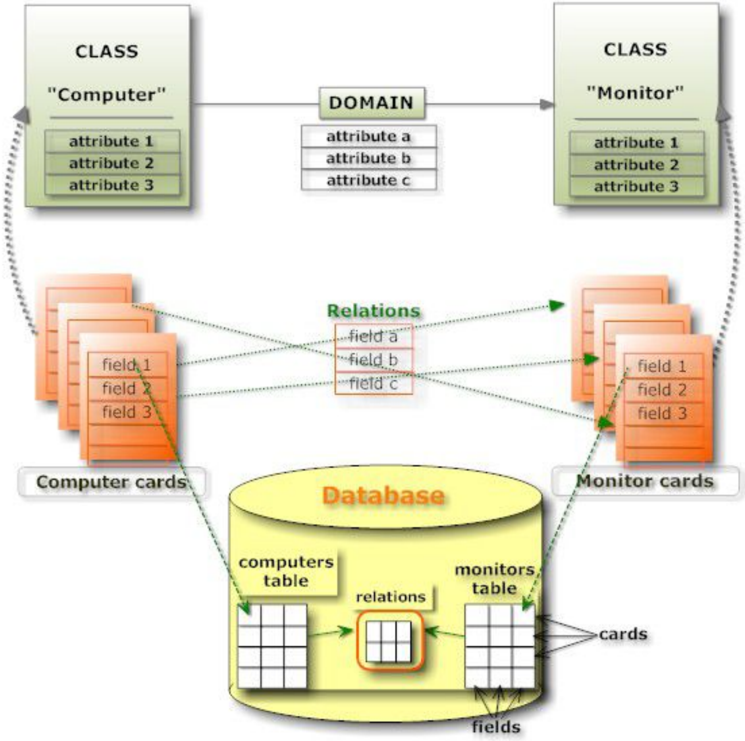
Start by:

1. Managing a complete and accurate set of objects and relationships
2. Extend the system once we have become more familiar with CMDBuild rules and usage.

Identify:

- **Types of items** to manage (classes):
  - IT assets (computers, peripherals, network systems, etc.)
  - assets related to real estates (buildings, plants, technical devices, etc.)
  - assets related to production plants (factories, plants, machines, etc.),
  - other types of assets (motor vehicles, etc.)

Data model



- **Type of item** (class) **attributes**:
  - useful to define each class (e.g. for an asset there will be a code, a description, supplier, etc.)
  - the related type of datum (string, long text, etc. date, "lookup" list, reference, geographical attribute, or open or closed polygon)
- relations between classes
- "attributes" useful to describe each **"domain"** (e.g. the role of each person in charge of a service, the type of dependency between two assets, etc.) and the related type of data (string, long text, integer..)
- user accounts for every class

Data model

Possible hierarchy of classes

- to define abstract classes (Superclasses) which can be used as templates (e.g. "Computer")
- derive subclasses (for example, "Desktop", "Laptop", "Server") which will include
  - real data
  - shared attributes (specified in the superclass) and the ones specified in the subclass
  - domain relations of the superclass and the specific domains.

It's important to identify a hierarchy that meets the current and future needs of the organization since a class can not be automatically converted into a superclass.

What do we want/need here?

Once the **entity-relationship model** has been defined, we have to define the individual classes and relatedattributes / data types.

At the end of this operation we should:

• use the Administration Module to model the system you've designed using E-R editor

• use the Management Module to insert, update and display cards

Ingest Excel Sheet

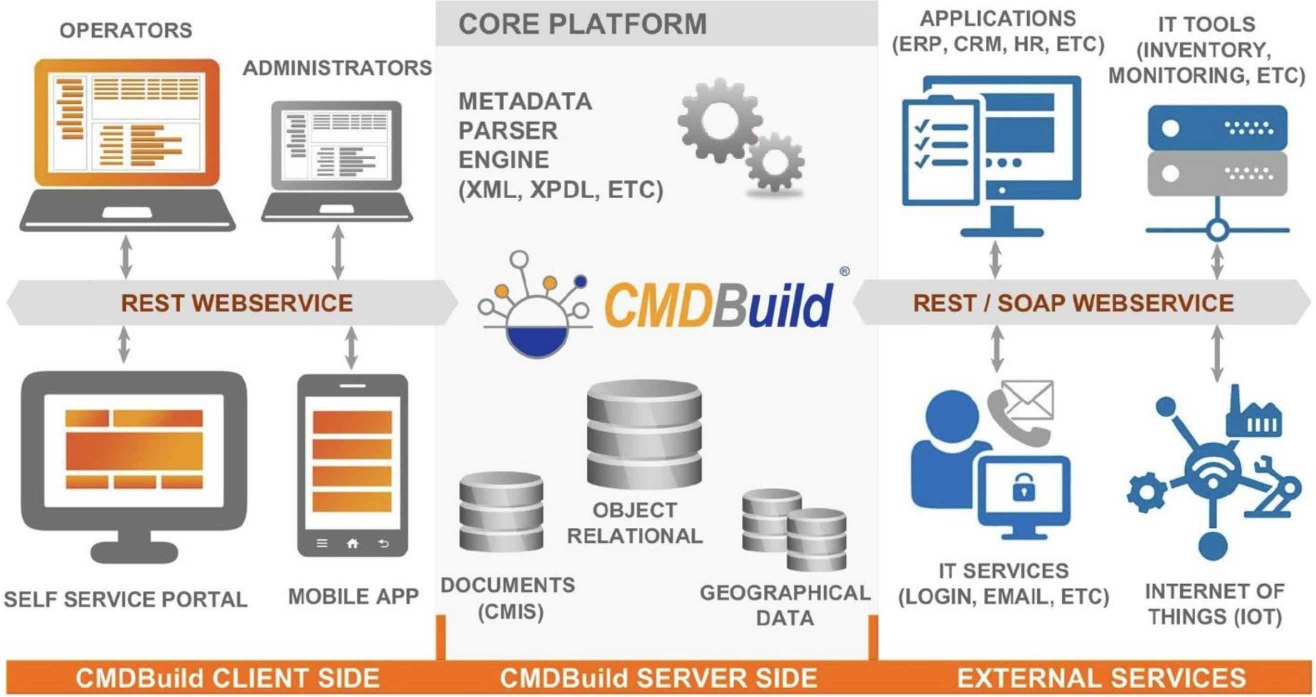
–

M. Rumore

Q&A

Backup Material

Predictive Maintenance


OPERATORS
ADMINISTRATORS
REST WEBSERVICE
SELF SERVICE PORTAL
MOBILE APP
CORE PLATFORM
METADATA PARSER ENGINE (XML, XPDL, ETC)
CMDBuild®
DOCUMENTS (CMIS)
OBJECT RELATIONAL
GEOGRAPHICAL DATA
APPLICATIONS (ERP, CRM, HR, ETC)
IT TOOLS (INVENTORY, MONITORING, ETC)
REST / SOAP WEBSERVICE
IT SERVICES (LOGIN, EMAIL, ETC)
INTERNET OF THINGS (IOT)
CMDBuild CLIENT SIDE
CMDBuild SERVER SIDE
EXTERNAL SERVICES