



IoT Platforms and Optimisation

Presented by:

Thorsten Ulbricht, Roche

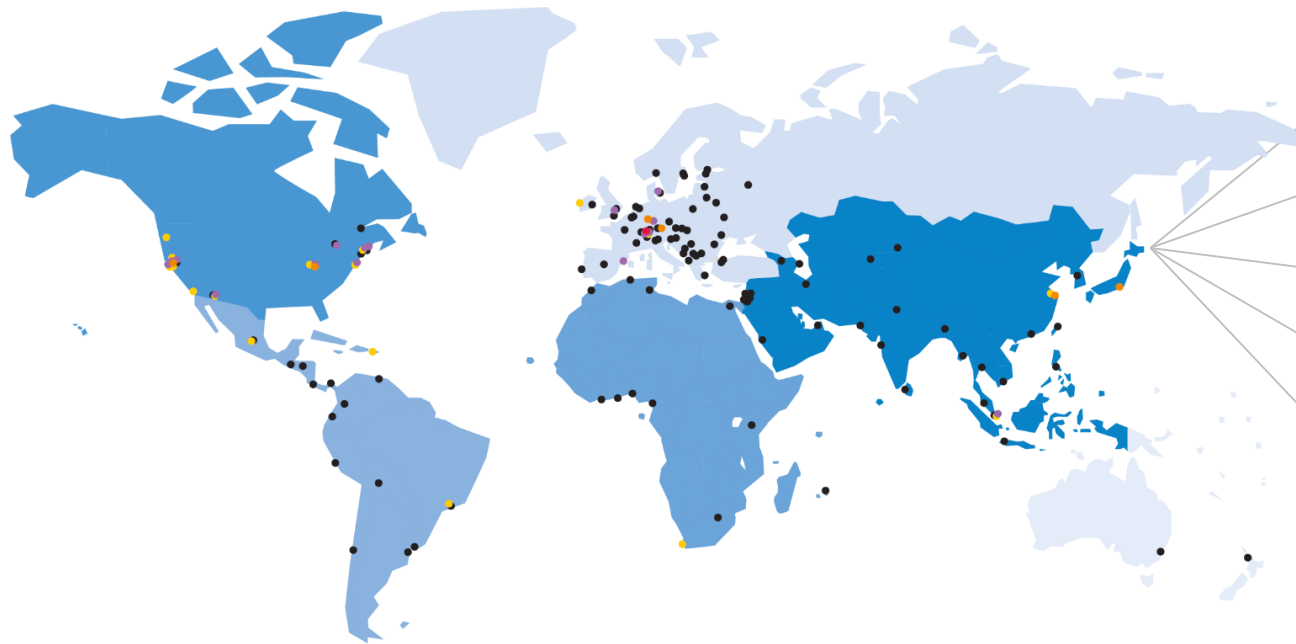
Philipp Sutter, Octavesoft GmbH

This is Roche

2019

Roche is a global pioneer in personalised healthcare

Innovation: it's in our DNA



CHF 11.0 billion R&D investment

30 R&D sites worldwide *

CHF 56.8 billion sales

26 manufacturing sites
worldwide

94,442 employees worldwide

#1 R&D investor in healthcare**

Among **top 10** R&D investors across
industries

***Doing
now
what
patients
need next***

We believe it's urgent to deliver medical solutions right now – even as we develop innovations for the future. We are passionate about transforming patients' lives. We are courageous in both decision and action. And we believe that good business means a better world.

That is why we come to work each day. We commit ourselves to scientific rigour, unassailable ethics, and access to medical innovations for all. We do this today to build a better tomorrow.

We are proud of who we are, what we do, and how we do it. We are many, working as one across functions, across companies, and across the world.

We are Roche.

The woman shown here was in the midst of receiving treatment for her cancer when photographed for the Roche Annual Report 2017 last year. Today, thanks to modern medicines and diagnostic tests, she is cancer-free and enjoying life again.



Creating sustainable value

Our impact on society



Innovating for patients

127 million patients treated with our medicines

32 Roche medicines on the WHO Model List of Essential Medicines

20 billion tests conducted with Roche products

Providing a great workplace

30% of key leadership positions now held by women

22% of key leaders with diverse work experience

72%* employee engagement rate

Being a trustworthy partner

23 new partnerships in Diagnostics

107 new partnerships in Pharmaceuticals

100% of approximately 1,000 business-critical suppliers risk-assessed

Protecting the environment & supporting communities

13% decrease in water consumption since 2015

23% improvement in the eco-balance since 2014

10% decrease in energy consumption since 2015

*measured by Global Employee Opinion Survey 2017

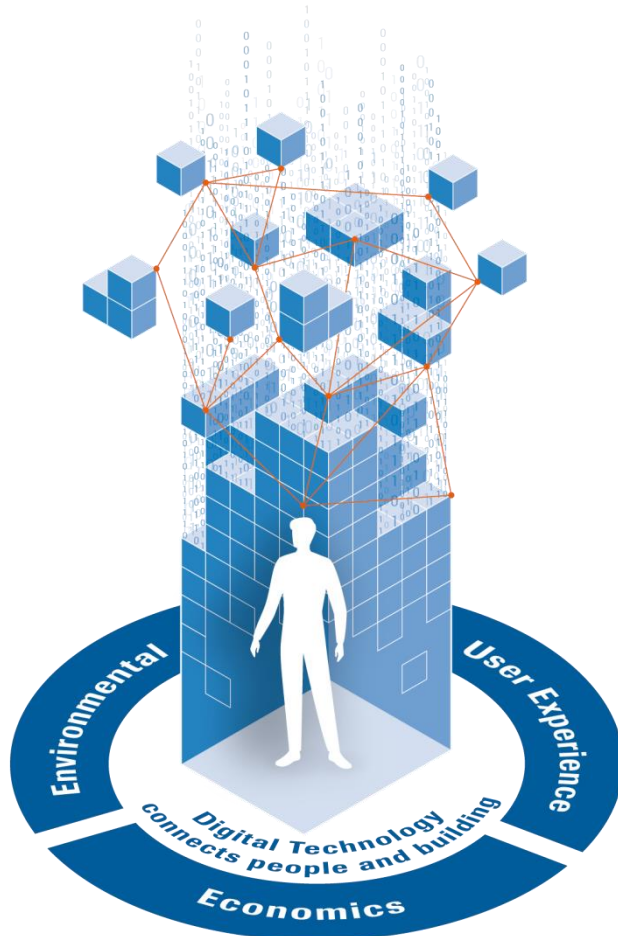
Smart Building IoT Platform and Optimisation

Philipp Sutter OctaveSoft GmbH, Thorsten Ulbricht Roche



Smart Building

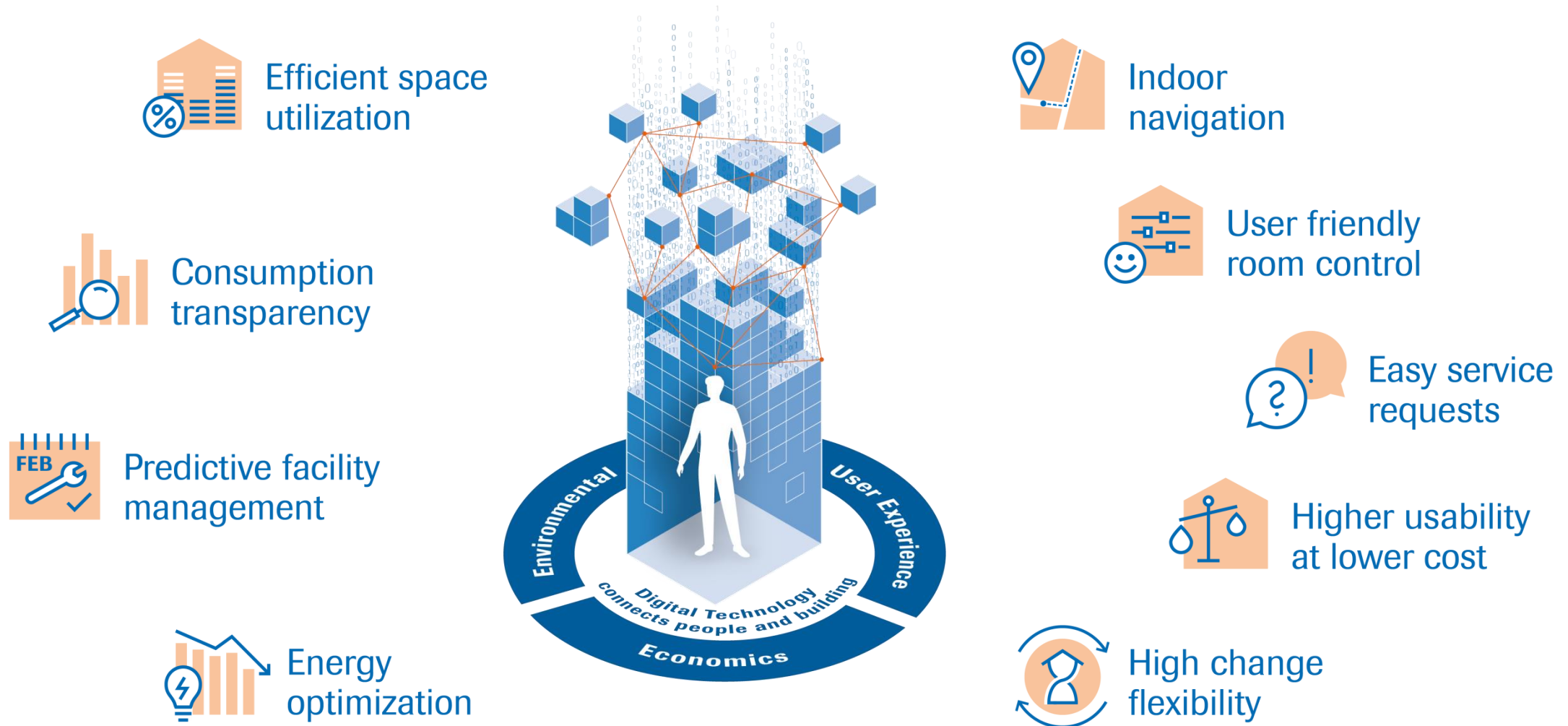
Vision of a Smart Building @ Roche



*Our **Smart Building** ...
is interconnected with its
environment,
is easily changeable in its capabilities,
provides a high user satisfaction and
displays sustainable values .*

Smart Building

Requirements to a Smart Building



Everyone knows...

....Data is the new gold

But no one can tell where to dig.

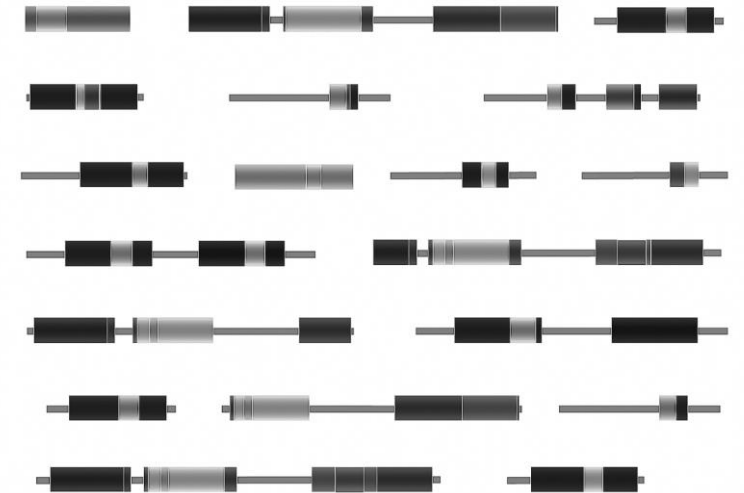
Just putting together data in an interdisciplinary approach can help to find new targets.



A few assumptions

to find valuable claims

- Everything we put in Energy, all kinds of Energy
- Things we do very often, all kinds of multipliers
- Unknown incidents, abnormalities in general
- Unknown relationships between processes
- Or to make things easier, a mix of everything mentioned above!

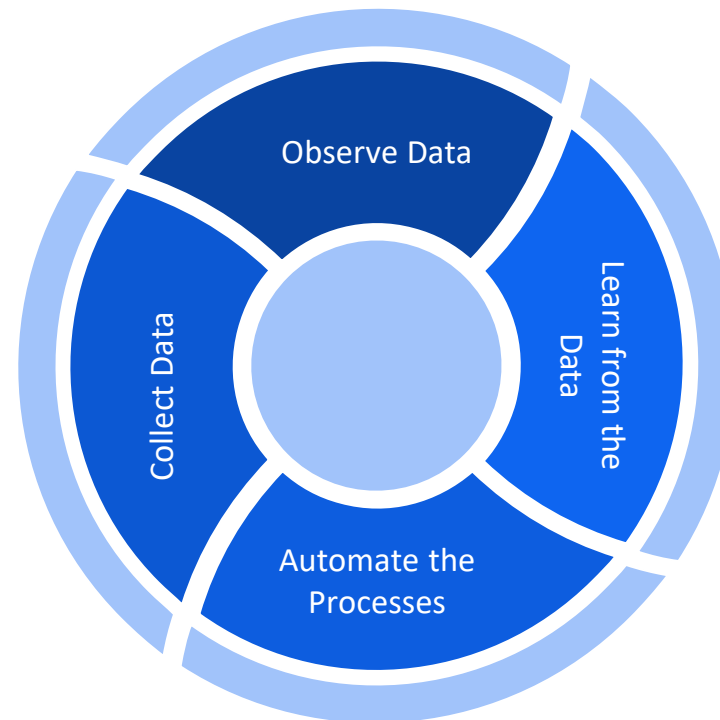


Why we need an IoT Platform

And what is it for...

An IoT platform is a multi-layer technology that enables straightforward provisioning, management, and automation of connected devices within the Internet of Things universe. Thus, an IoT platform can be wearing different hats depending on how you look at it.

found on the Internet of Humans



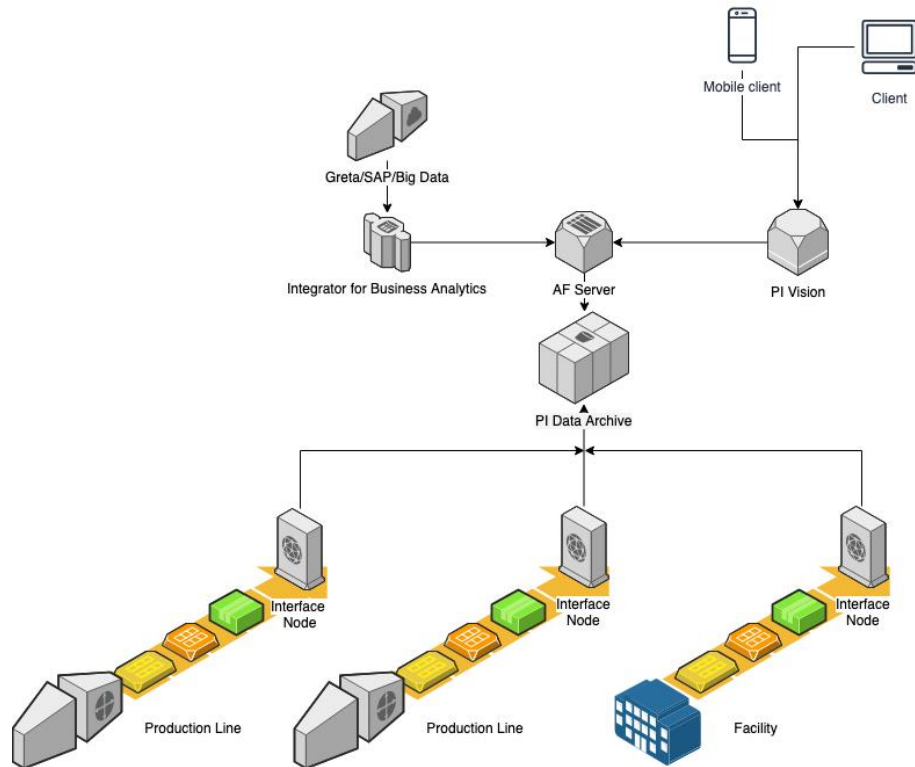
Establishing an IOT Platform is a journey. The data created from processes needs to get visible. First if the meaning of the data is revealed and a continuous flow is established, processes can be automated.



Analyzing

to get an overview things need to be looked at

To get a further insight on collected data, especially when it is cross platform, interdisciplinary data, best is to have this data fast accessed. Multi interfacing systems preferred.



Controlling

gaining control over the systems that create the data

After knowing what is going on, it is now time to find triggers and functions from a process point of view.

- Interfaces for external controls
- Latency or other restrictions
- Network connectivity



Establish new control circuits

With integrated real time data and established controls over processes it is now possible to create cross platform control circuits.

- Intelligent systems directly react on new circumstances.
- Values from measured values can trigger actions for actors over multiple systems



Use Case 1 Building 098 Basel

B098 IVR (2023)

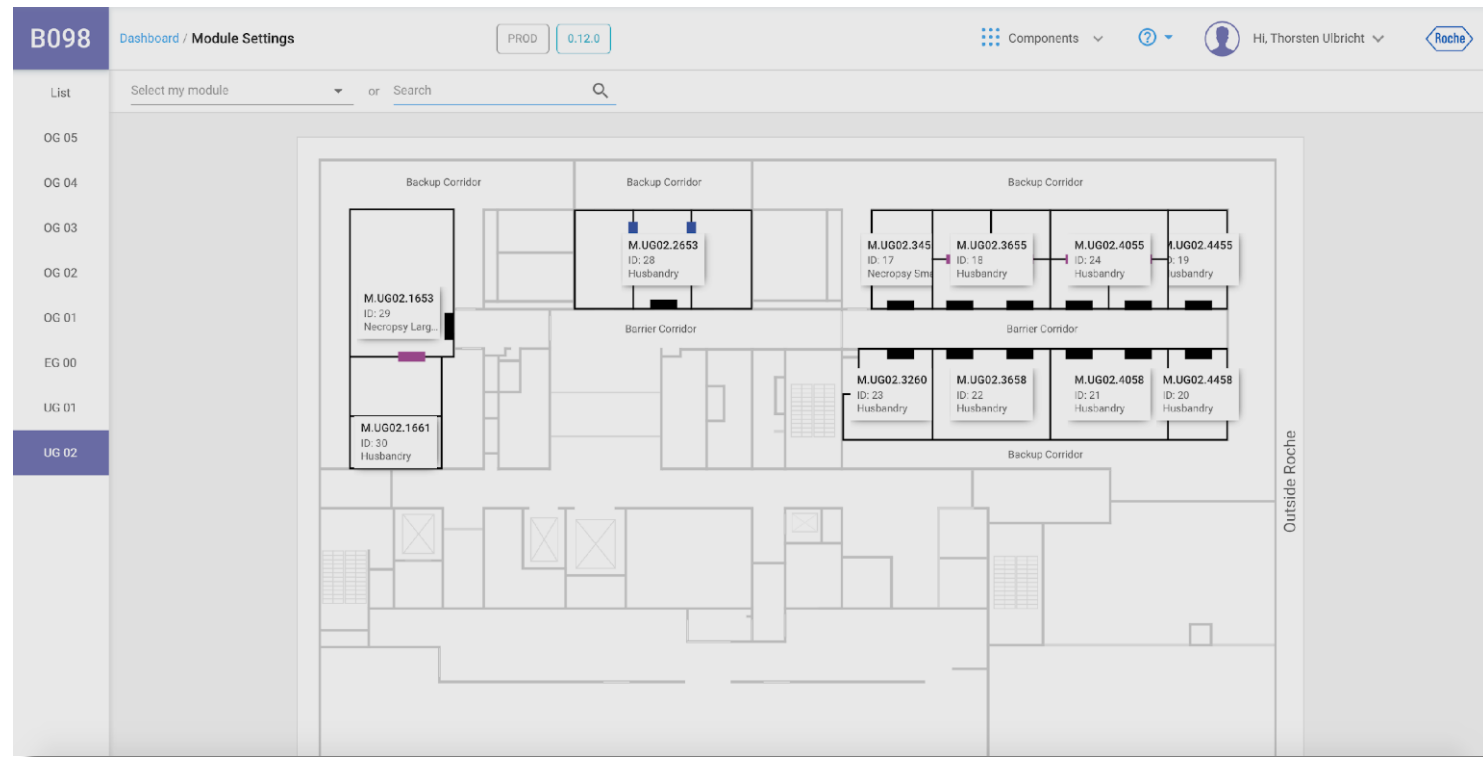
We would call this a extrinsic use case. Knowledge about what is going on inside a facility helps us to find potential savings within another system.



Business Case 1

Building B098 Basel

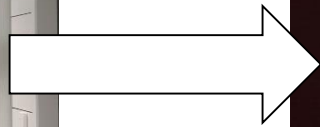
- Inside Building 098, Roche is operating 120 static Tablet PC devices. These devices serve as Doorplate, but also as a remote control for all building functions assigned to a room.



Business Case 1

Building B098 Basel

Situation at the site Bldg 098



- In order to extend the lifetime of this devices and reduce the energy consumption of the building, we developed the following.

If the counter is
>0 the endpoint
value is true.

On false, an app switches the devices off, on true, a simulated mouse/touch event wakes them up.



```

1  return InterferenceInjection.isInterfering ? return new InterferenceInjection\(\) : null;
2
3  @Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
4      return inflater.inflate(R.layout.activity_main, container, false);
5  }
6
7  @Override public void onViewCreated(View view, Bundle savedInstanceState) {
8      // Inflate the layout for this activity
9      setContentView(R.layout.activity_main);
10
11      // Find the button
12      TextView textView = (TextView) findViewById(R.id.textView);
13
14      // Set the text of the button
15      textView.setText("Click Me");
16
17      // Set the click listener
18      textView.setOnClickListener(new View.OnClickListener() {
19          @Override public void onClick(View v) {
20              // Show a toast message
21              Toast.makeText(getApplicationContext(), "Button Clicked", Toast.LENGTH_SHORT).show();
22          }
23      });
24  }
25
26  // Method to show a toast message
27  private void showToast(String message) {
28      Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
29  }
30
31  // Method to show a dialog box
32  private void showDialog() {
33      AlertDialog.Builder builder = new AlertDialog.Builder(this);
34      builder.setTitle("Dialog Box");
35      builder.setMessage("This is a dialog box message.");
36      builder.setPositiveButton("OK", null);
37      builder.setNegativeButton("Cancel", null);
38      builder.create().show();
39  }
40
41  // Method to show a progress bar
42  private void showProgressBar() {
43      ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
44      progressBar.setVisibility(View.VISIBLE);
45      // Simulate a long task
46      try {
47          Thread.sleep(2000);
48      } catch (InterruptedException e) {
49          e.printStackTrace();
50      }
51      progressBar.setVisibility(View.GONE);
52  }
53
54  // Method to show a spinner
55  private void showSpinner() {
56      Spinner spinner = (Spinner) findViewById(R.id.spinner);
57      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, new String[]{"Apple", "Banana", "Mango", "Orange"});
58      adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
59      spinner.setAdapter(adapter);
60  }
61
62  // Method to show a list view
63  private void showListView() {
64      ListView listView = (ListView) findViewById(R.id.listView);
65      ArrayList<String> data = new ArrayList<String>();
66      data.add("Item 1");
67      data.add("Item 2");
68      data.add("Item 3");
69      data.add("Item 4");
70      data.add("Item 5");
71      listView.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, data));
72  }
73
74  // Method to show a RecyclerView
75  private void showRecyclerView() {
76      RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
77      ArrayList<String> data = new ArrayList<String>();
78      data.add("Item 1");
79      data.add("Item 2");
80      data.add("Item 3");
81      data.add("Item 4");
82      data.add("Item 5");
83      recyclerView.setAdapter(new RecyclerView.Adapter<ViewHolder>() {
84          @Override public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
85              View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_view, parent, false);
86              return new ViewHolder(view);
87          }
88          @Override public void onBindViewHolder(ViewHolder holder, int position) {
89              holder.textView.setText(data.get(position));
90          }
91          @Override public int getItemCount() {
92              return data.size();
93          }
94      });
95  }
96
97  // Method to show a ViewPager
98  private void showViewPager() {
99      ViewPager viewPager = (ViewPager) findViewById(R.id.viewPager);
100     ArrayList<String> data = new ArrayList<String>();
101     data.add("Page 1");
102     data.add("Page 2");
103     data.add("Page 3");
104     data.add("Page 4");
105     viewPager.setAdapter(new ViewPager.Adapter<String>() {
106         @Override public String getItem(int position) {
107             return data.get(position);
108         }
109         @Override public int getCount() {
110             return data.size();
111         }
112     });
113 }
114
115 // Method to show a ScrollView
116 private void showScrollView() {
117     ScrollView scrollView = (ScrollView) findViewById(R.id.scrollView);
118     TextView textView = (TextView) findViewById(R.id.textView);
119     textView.setText("This is a scroll view message.");
120     scrollView.addView(textView);
121 }
122
123 // Method to show a TabLayout
124 private void showTabLayout() {
125     TabLayout tabLayout = (TabLayout) findViewById(R.id.tabLayout);
126     TextView textView = (TextView) findViewById(R.id.textView);
127     textView.setText("This is a tab layout message.");
128     tabLayout.addTab(new TabLayout.Tab().setText("Tab 1"));
129     tabLayout.addTab(new TabLayout.Tab().setText("Tab 2"));
130     tabLayout.select(0);
131 }
132
133 // Method to show a BottomNavigationView
134 private void showBottomNavigationView() {
135     BottomNavigationView bottomNavigationView = (BottomNavigationView) findViewById(R.id.bottomNavigationView);
136     bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
137         @Override public boolean onNavigationItemSelected(@NonNull MenuItem item) {
138             return true;
139         }
140     });
141     bottomNavigationView.getItem(0).setChecked(true);
142 }
143
144 // Method to show a DrawerLayout
145 private void showDrawerLayout() {
146     DrawerLayout drawerLayout = (DrawerLayout) findViewById(R.id.drawerLayout);
147     TextView textView = (TextView) findViewById(R.id.textView);
148     textView.setText("This is a drawer layout message.");
149     drawerLayout.addDrawerView(new DrawerLayout.DrawerListener() {
150         @Override public void onDrawerSlide(View drawerView, float slideOffset) {
151             // Do nothing
152         }
153         @Override public void onDrawerOpen(View drawerView) {
154             // Do nothing
155         }
156         @Override public void onDrawerClose(View drawerView) {
157             // Do nothing
158         }
159     });
160 }
161
162 // Method to show a CardView
163 private void showCardView() {
164     CardView cardView = (CardView) findViewById(R.id.cardView);
165     TextView textView = (TextView) findViewById(R.id.textView);
166     textView.setText("This is a card view message.");
167     cardView.setCardElevation(10f);
168 }
169
170 // Method to show a FloatingActionButton
171 private void showFloatingActionButton() {
172     FloatingActionButton floatingActionButton = (FloatingActionButton) findViewById(R.id.floatingActionButton);
173     floatingActionButton.setOnClickListener(new View.OnClickListener() {
174         @Override public void onClick(View v) {
175             // Show a toast message
176             showToast("Floating Action Button Clicked");
177         }
178     });
179 }
180
181 // Method to show a Snackbar
182 private void showSnackbar() {
183     Snackbar snackbar = Snackbar.make(findViewById(R.id.snackbar_text), "This is a snackbar message.", Snackbar.LENGTH_LONG);
184     snackbar.show();
185 }
186
187 // Method to show a BottomSheetDialog
188 private void showBottomSheetDialog() {
189     BottomSheetDialog dialog = new BottomSheetDialog(this);
190     TextView textView = (TextView) findViewById(R.id.textView);
191     textView.setText("This is a bottom sheet dialog message.");
192     dialog.addContentView(textView, new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT));
193     dialog.show();
194 }
195
196 // Method to show a DialogFragment
197 private void showDialogFragment() {
198     DialogFragment dialogFragment = new DialogFragment();
199     dialogFragment.show(getSupportFragmentManager(), "DialogFragment");
200 }
201
202 // Method to show a AlertDialog
203 private void showAlertDialog() {
204     AlertDialog.Builder builder = new AlertDialog.Builder(this);
205     builder.setTitle("AlertDialog");
206     builder.setMessage("This is an alert dialog message.");
207     builder.setPositiveButton("OK", null);
208     builder.setNegativeButton("Cancel", null);
209     builder.create().show();
210 }
211
212 // Method to show a ProgressDialog
213 private void showProgressDialog() {
214     ProgressDialog progressDialog = new ProgressDialog(this);
215     progressDialog.setTitle("ProgressDialog");
216     progressDialog.setMessage("This is a progress dialog message.");
217     progressDialog.show();
218 }
219
220 // Method to show a AlertDialog.Builder
221 private void showAlertDialogBuilder() {
222     AlertDialog.Builder builder = new AlertDialog.Builder(this);
223     builder.setTitle("AlertDialog.Builder");
224     builder.setMessage("This is an alert dialog message.");
225     builder.setPositiveButton("OK", null);
226     builder.setNegativeButton("Cancel", null);
227     builder.create().show();
228 }
229
230 // Method to show a AlertDialog.Builder
231 private void showAlertDialogBuilder() {
232     AlertDialog.Builder builder = new AlertDialog.Builder(this);
233     builder.setTitle("AlertDialog.Builder");
234     builder.setMessage("This is an alert dialog message.");
235     builder.setPositiveButton("OK", null);
236     builder.setNegativeButton("Cancel", null);
237     builder.create().show();
238 }
239
240 // Method to show a AlertDialog.Builder
241 private void showAlertDialogBuilder() {
242     AlertDialog.Builder builder = new AlertDialog.Builder(this);
243     builder.setTitle("AlertDialog.Builder");
244     builder.setMessage("This is an alert dialog message.");
245     builder.setPositiveButton("OK", null);
246     builder.setNegativeButton("Cancel", null);
247     builder.create().show();
248 }
249
250 // Method to show a AlertDialog.Builder
251 private void showAlertDialogBuilder() {
252     AlertDialog.Builder builder = new AlertDialog.Builder(this);
253     builder.setTitle("AlertDialog.Builder");
254     builder.setMessage("This is an alert dialog message.");
255     builder.setPositiveButton("OK", null);
256     builder.setNegativeButton("Cancel", null);
257     builder.create().show();
258 }
259
260 // Method to show a AlertDialog.Builder
261 private void showAlertDialogBuilder() {
262     AlertDialog.Builder builder = new AlertDialog.Builder(this);
263     builder.setTitle("AlertDialog.Builder");
264     builder.setMessage("This is an alert dialog message.");
265     builder.setPositiveButton("OK", null);
266     builder.setNegativeButton("Cancel", null);
267     builder.create().show();
268 }
269
270 // Method to show a AlertDialog.Builder
271 private void showAlertDialogBuilder() {
272     AlertDialog.Builder builder = new AlertDialog.Builder(this);
273     builder.setTitle("AlertDialog.Builder");
274     builder.setMessage("This is an alert dialog message.");
275     builder.setPositiveButton("OK", null);
276     builder.setNegativeButton("Cancel", null);
277     builder.create().show();
278 }
279
280 // Method to show a AlertDialog.Builder
281 private void showAlertDialogBuilder() {
282     AlertDialog.Builder builder = new AlertDialog.Builder(this);
283     builder.setTitle("AlertDialog.Builder");
284     builder.setMessage("This is an alert dialog message.");
285     builder.setPositiveButton("OK", null);
286     builder.setNegativeButton("Cancel", null);
287     builder.create().show();
288 }
289
290 // Method to show a AlertDialog.Builder
291 private void showAlertDialogBuilder() {
292     AlertDialog.Builder builder = new AlertDialog.Builder(this);
293     builder.setTitle("AlertDialog.Builder");
294     builder.setMessage("This is an alert dialog message.");
295     builder.setPositiveButton("OK", null);
296     builder.setNegativeButton("Cancel", null);
297     builder.create().show();
298 }
299
300 // Method to show a AlertDialog.Builder
301 private void showAlertDialogBuilder() {
302     AlertDialog.Builder builder = new AlertDialog.Builder(this);
303     builder.setTitle("AlertDialog.Builder");
304     builder.setMessage("This is an alert dialog message.");
305     builder.setPositiveButton("OK", null);
306     builder.setNegativeButton("Cancel", null);
307     builder.create().show();
308 }
309
310 // Method to show a AlertDialog.Builder
311 private void showAlertDialogBuilder() {
312     AlertDialog.Builder builder = new AlertDialog.Builder(this);
313     builder.setTitle("AlertDialog.Builder");
314     builder.setMessage("This is an alert dialog message.");
315     builder.setPositiveButton("OK", null);
316     builder.setNegativeButton("Cancel", null);
317     builder.create().show();
318 }
319
320 // Method to show a AlertDialog.Builder
321 private void showAlertDialogBuilder() {
322     AlertDialog.Builder builder = new AlertDialog.Builder(this);
323     builder.setTitle("AlertDialog.Builder");
324     builder.setMessage("This is an alert dialog message.");
325     builder.setPositiveButton("OK", null);
326     builder.setNegativeButton("Cancel", null);
327     builder.create().show();
328 }
329
330 // Method to show a AlertDialog.Builder
331 private void showAlertDialogBuilder() {
332     AlertDialog.Builder builder = new AlertDialog.Builder(this);
333     builder.setTitle("AlertDialog.Builder");
334     builder.setMessage("This is an alert dialog message.");
335     builder.setPositiveButton("OK", null);
336     builder.setNegativeButton("Cancel", null);
337     builder.create().show();
338 }
339
340 // Method to show a AlertDialog.Builder
341 private void showAlertDialogBuilder() {
342     AlertDialog.Builder builder = new AlertDialog.Builder(this);
343     builder.setTitle("AlertDialog.Builder");
344     builder.setMessage("This is an alert dialog message.");
345     builder.setPositiveButton("OK", null);
346     builder.setNegativeButton("Cancel", null);
347     builder.create().show();
348 }
349
350 // Method to show a AlertDialog.Builder
351 private void showAlertDialogBuilder() {
352     AlertDialog.Builder builder = new AlertDialog.Builder(this);
353     builder.setTitle("AlertDialog.Builder");
354     builder.setMessage("This is an alert dialog message.");
355     builder.setPositiveButton("OK", null);
356     builder.setNegativeButton("Cancel", null);
357     builder.create().show();
358 }
359
360 // Method to show a AlertDialog.Builder
361 private void showAlertDialogBuilder() {
362     AlertDialog.Builder builder = new AlertDialog.Builder(this);
363     builder.setTitle("AlertDialog.Builder");
364     builder.setMessage("This is an alert dialog message.");
365     builder.setPositiveButton("OK", null);
366     builder.setNegativeButton("Cancel", null);
367     builder.create().show();
368 }
369
370 // Method to show a AlertDialog.Builder
371 private void showAlertDialogBuilder() {
372     AlertDialog.Builder builder = new AlertDialog.Builder(this);
373     builder.setTitle("AlertDialog.Builder");
374     builder.setMessage("This is an alert dialog message.");
375     builder.setPositiveButton("OK", null);
376     builder.setNegativeButton("Cancel", null);
377     builder.create().show();
378 }
379
380 // Method to show a AlertDialog.Builder
381 private void showAlertDialogBuilder() {
382     AlertDialog.Builder builder = new AlertDialog.Builder(this);
38
```

Doing the Math

Smart Device Energy Savings

Example B098 Roompanel Devices

Energy

- $120 \text{ Devices} \times 45 \text{ Watt} \times 16 \text{ Hours Standby Time} \times 365 \text{ Days} = 31536 \text{ KWh}$
- Price Roche per KWh = 11Rp
- **$31536 \times 11 / 100 = 3468.96 \text{ CHF / Year}$**

Doing the Math

Smart Device Lifecycle Extension

Example B098 Roompanel Devices

Lifecycle

- Usual Device End of Life Timespan 3 Years
- Enhanced End of Life Timespan 5 Years
- Price Per Device 1'000 CHF = 200 CHF/Y enhanced
- Price per Device Deploy and install 2 M/h a 120 CHF = 240 CHF

Savings $120 \times 200 + 120 \times 240 = 52'800$ CHF (5 Years) = 10'560 CHF/Y

Possible savings

Example Smart Device Lifecycle & Energy

Example B098 Roompanel Devices

Overall per Year

- Energy 3'468.96 CHF
- Lifecycle 10'560.00 CHF
- -----
- **Total 14'028.96 CHF**

**Potential
Savings:**
14'000 CHF/
Year

Time Machine

Why is the data put together like written with a typewriter in the 70's?

Cause:

If you have your data in place and you know what is going on, all you need is a notepad and a calculator to find potential savings!

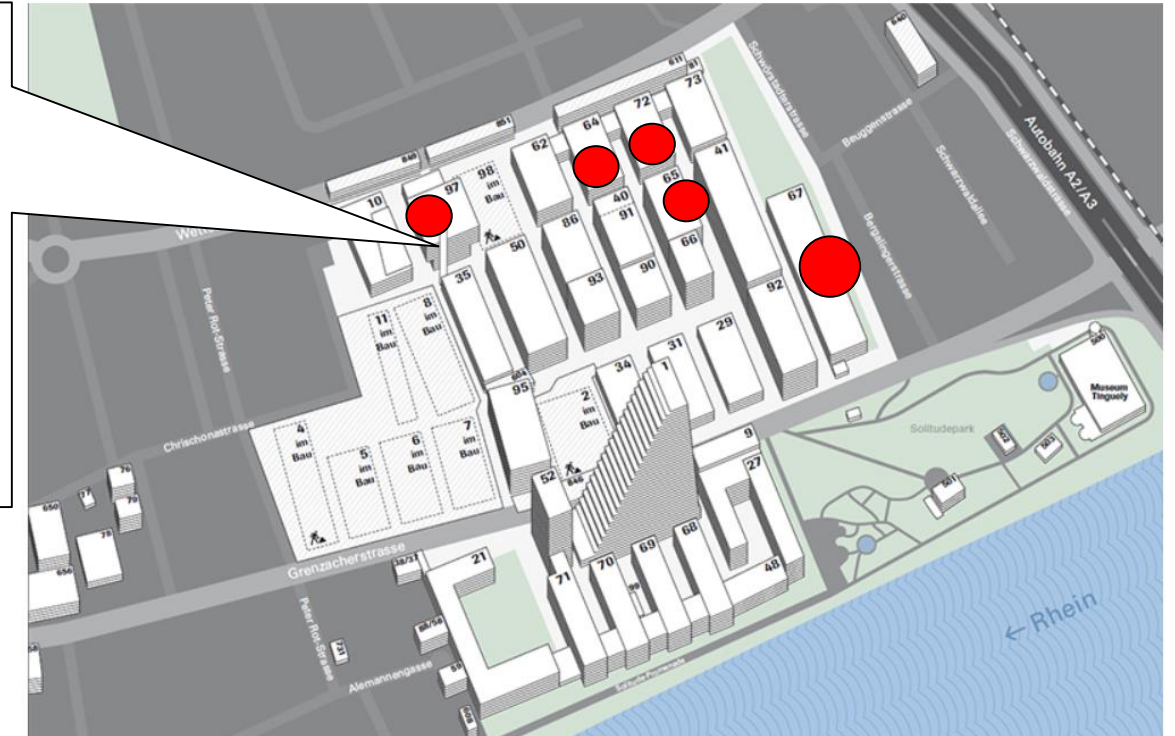
Business Case 2

«Crawling the Campus»

We would call this an intrinsic use case. Knowledge about what is going on inside a system directly helps us to find potential savings within the consumed power of the facility directly.

Asking around, we found complete Data of HVAC Systems for the Buildings 67,72,64,97 on the Roche Basel Campus. With the advanced analysis Tool Seeq, we started analysing possible savings on the data of the last 18 Month of building 67.

The potential findings were abstracted, and after harmonising the building data, the more comfortable search for cost savings began with Seeq....



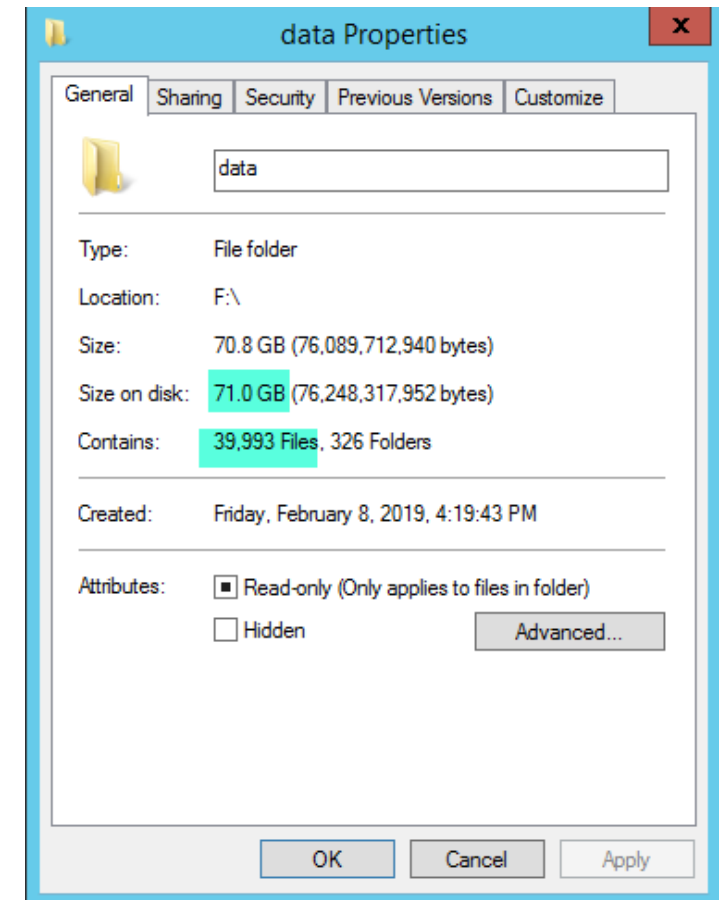
Integration of Data into PI

There is more data available...

- Building 098 is the most advanced “smart building” on the campus right now, so it was already fully connected, and all data were available in PI in real-time.
- But there are a lot more buildings on the campus, some of them rather old, but all of them with some form of HVAC Systems.
- Building managers say they could provide data in the form of
 - CSV files
 - EXCEL files

Ask, and it shall be given to you

➔ IDEA: use the PI UFL Connector to get this data into PI as well



Format of the Raw data

Data in the files typically was quite nicely structured, and could be used to

- automatically create PI tags
- automatically create AF elements
- import the data into PI

```

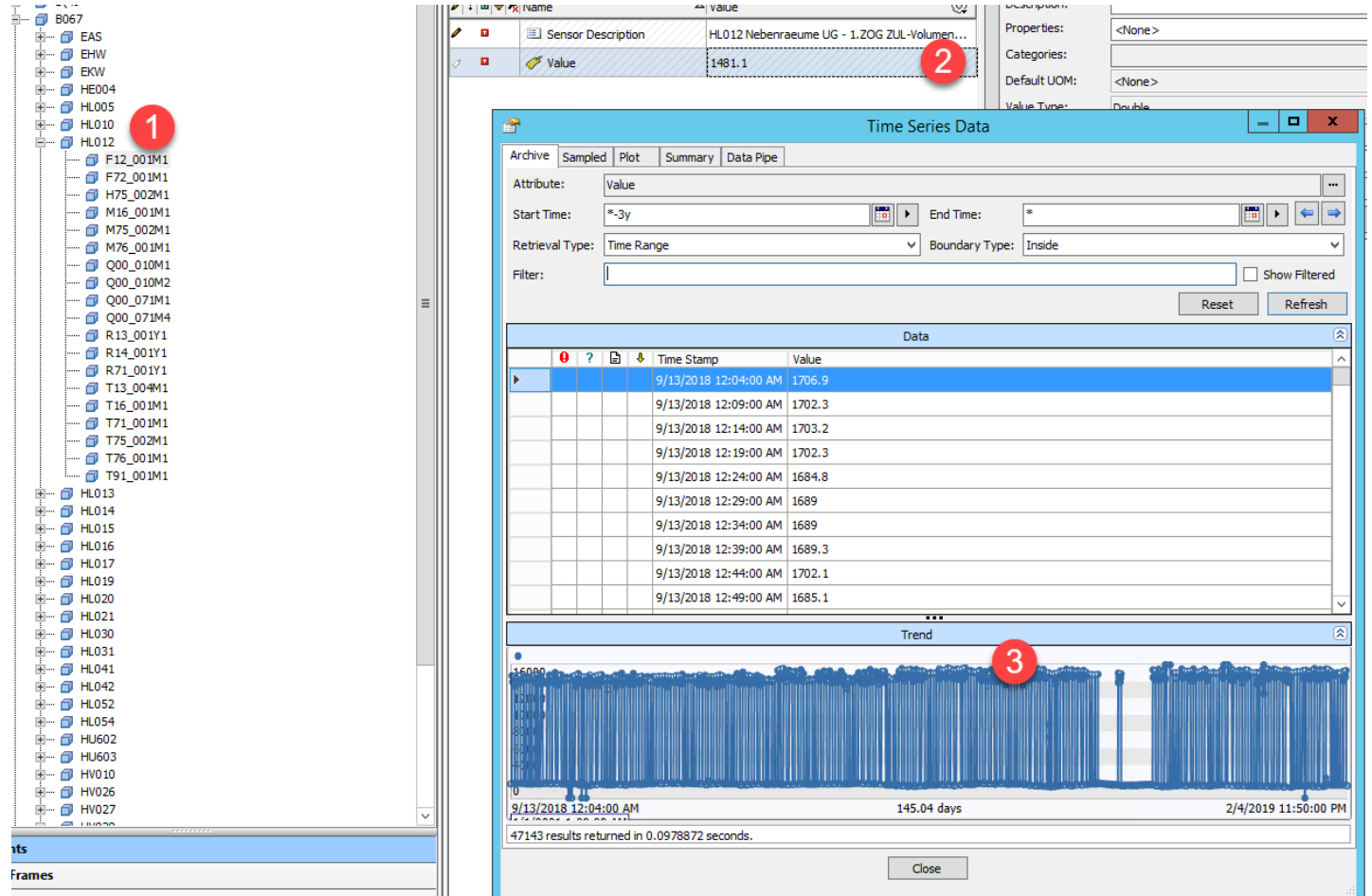
1 TAG;Description;Date;Time;Value
2 ---;-----;----;----;-----
3 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:04;514.67
4 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:09;531.38
5 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:14;509.85
6 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:19;514.38
7 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:24;534.21
8 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:29;488.04
9 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:34;488.32
10 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:39;539.31
11 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:44;502.20
12 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:49;551.49
13 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:54;509.29
14 067HL012F12_001M1;HL012 Nebenraeume UG - 1.ZOG ZUL-Volumenstrom (0-160000 m3/h) ;2017/02/23;00:59;483.51

```

PI UFL Connector results

UFL connector automatically created

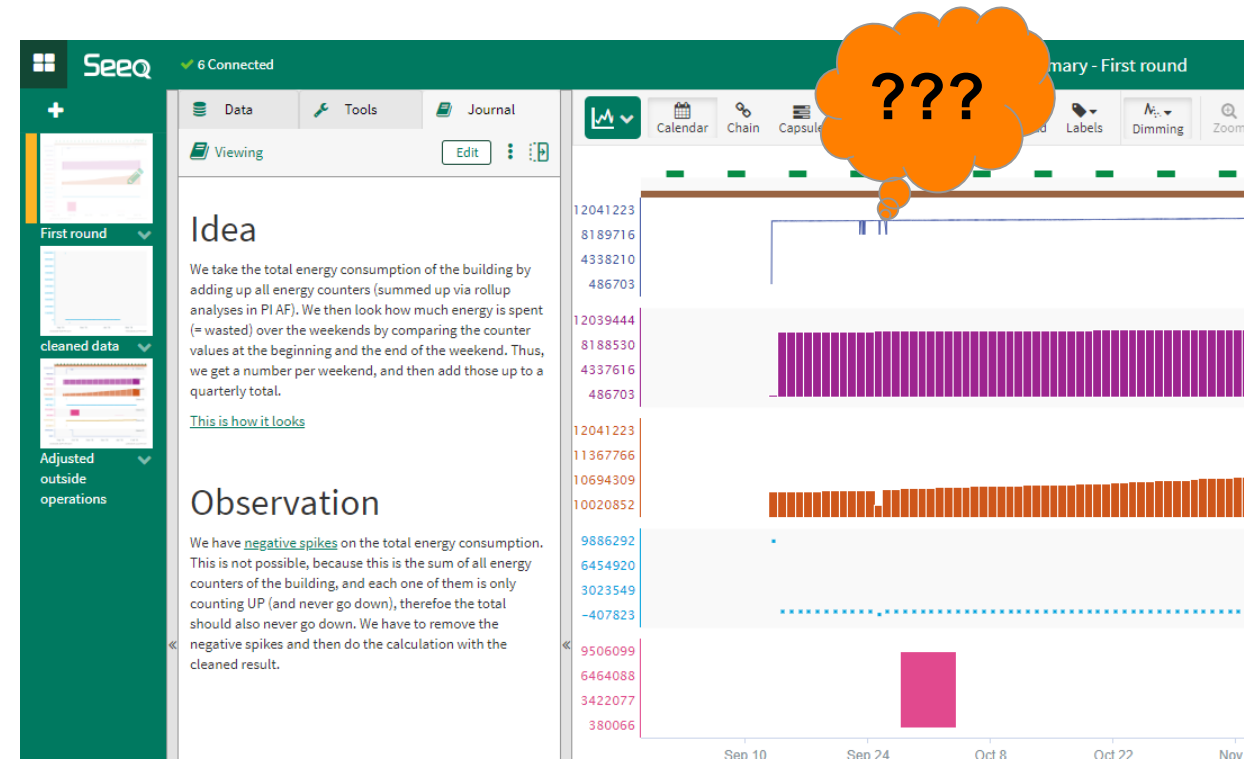
1. AF hierarchy and elements
2. PI tags including references to attributes
3. imported data



Analysis of the imported data using Seeq

Seeq experiences:

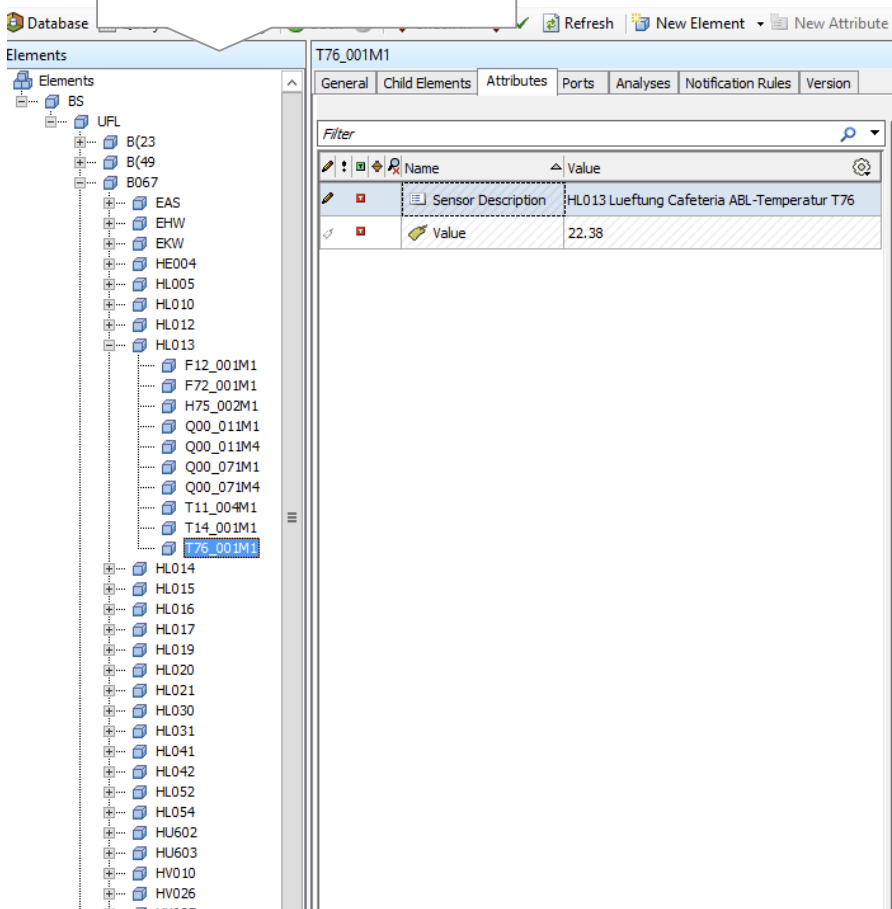
- very easy installation and connection to PI and AF
- graphical data exploration
- ideal tool to “play” with the data and easily find anomalies etc.
- run experiments and document the experiment results in the built-in Journal



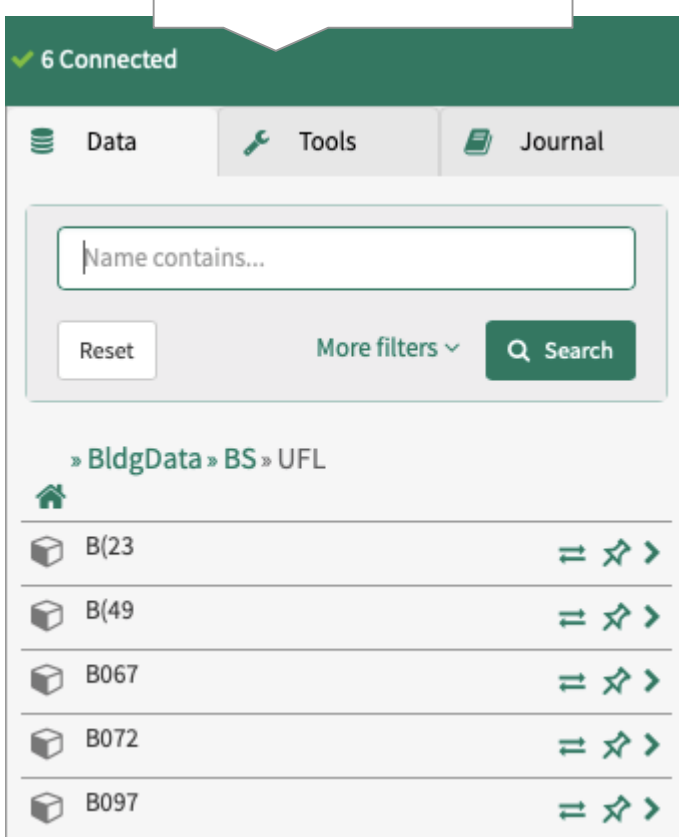
Business Case 2

Collect the Data

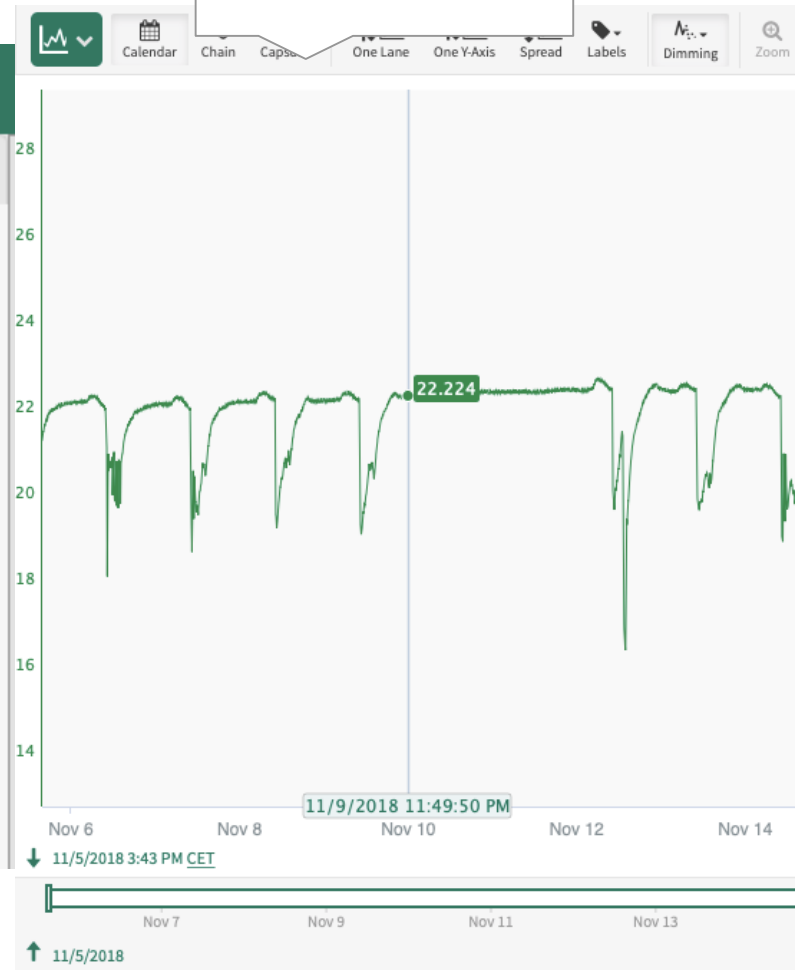
Throwing the data on a PI System, pulling it over to AF



connecting Seeq

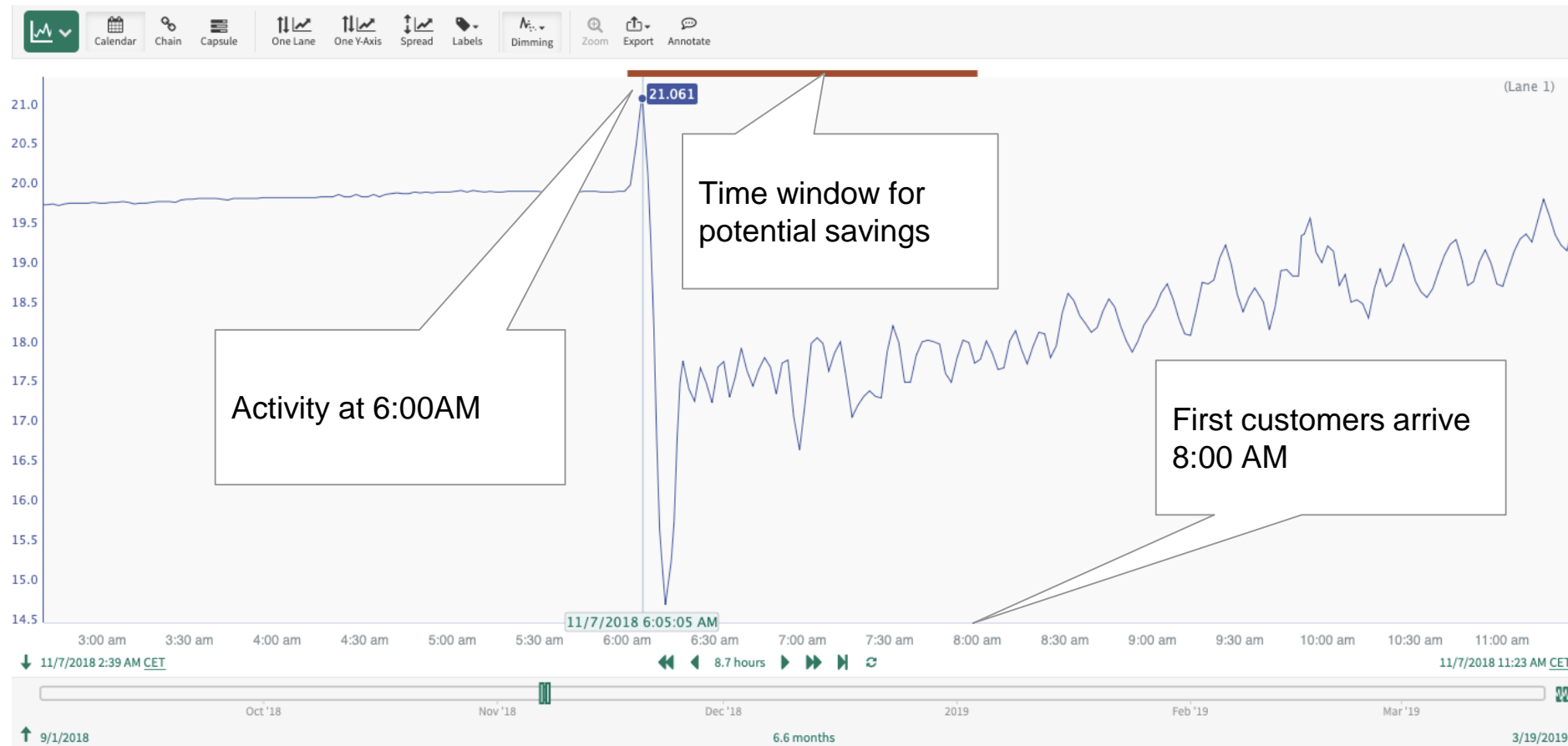


having a look



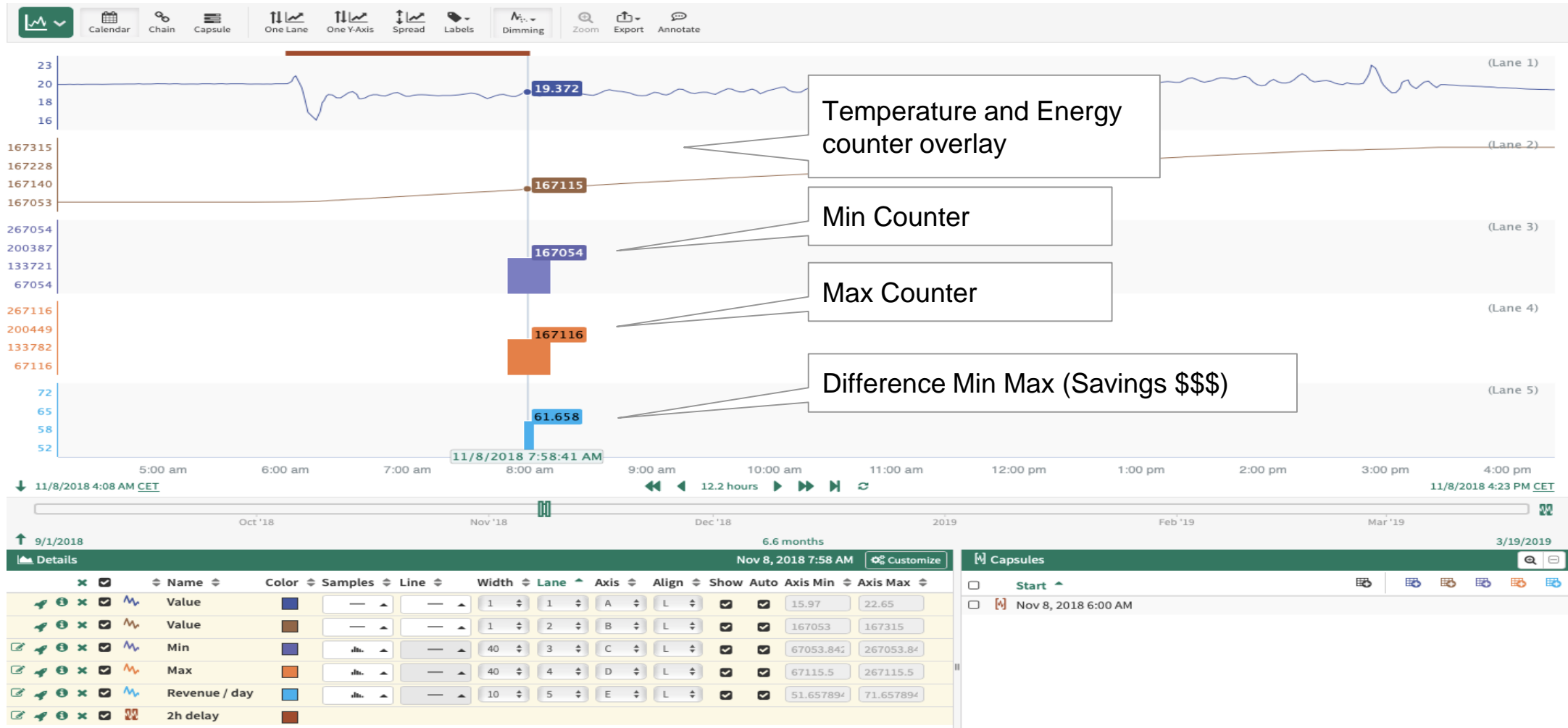
Business Case 2

Observe the Data



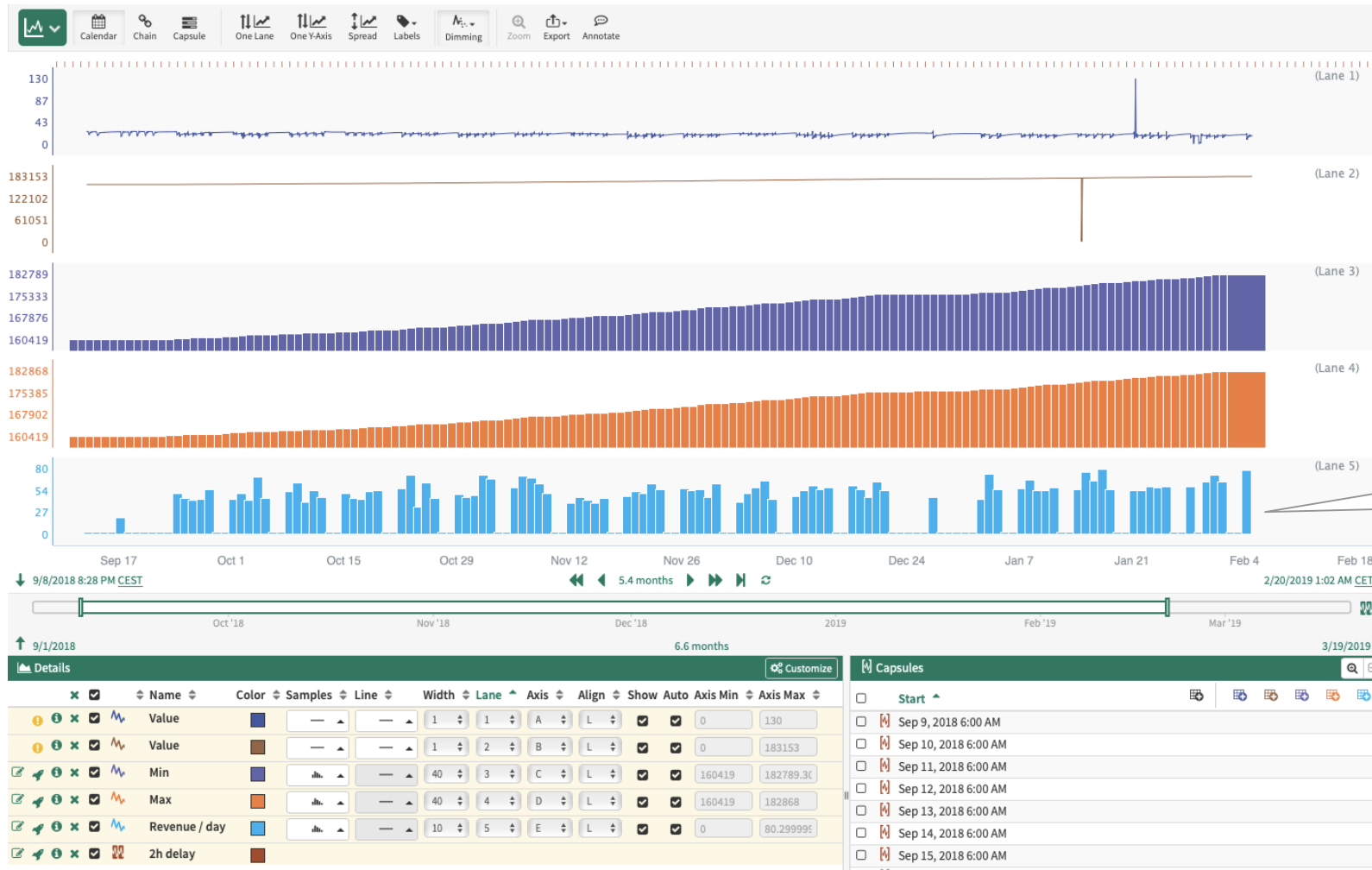
Business Case 2

Learning from the Data



Business Case 2

Learning from the Data

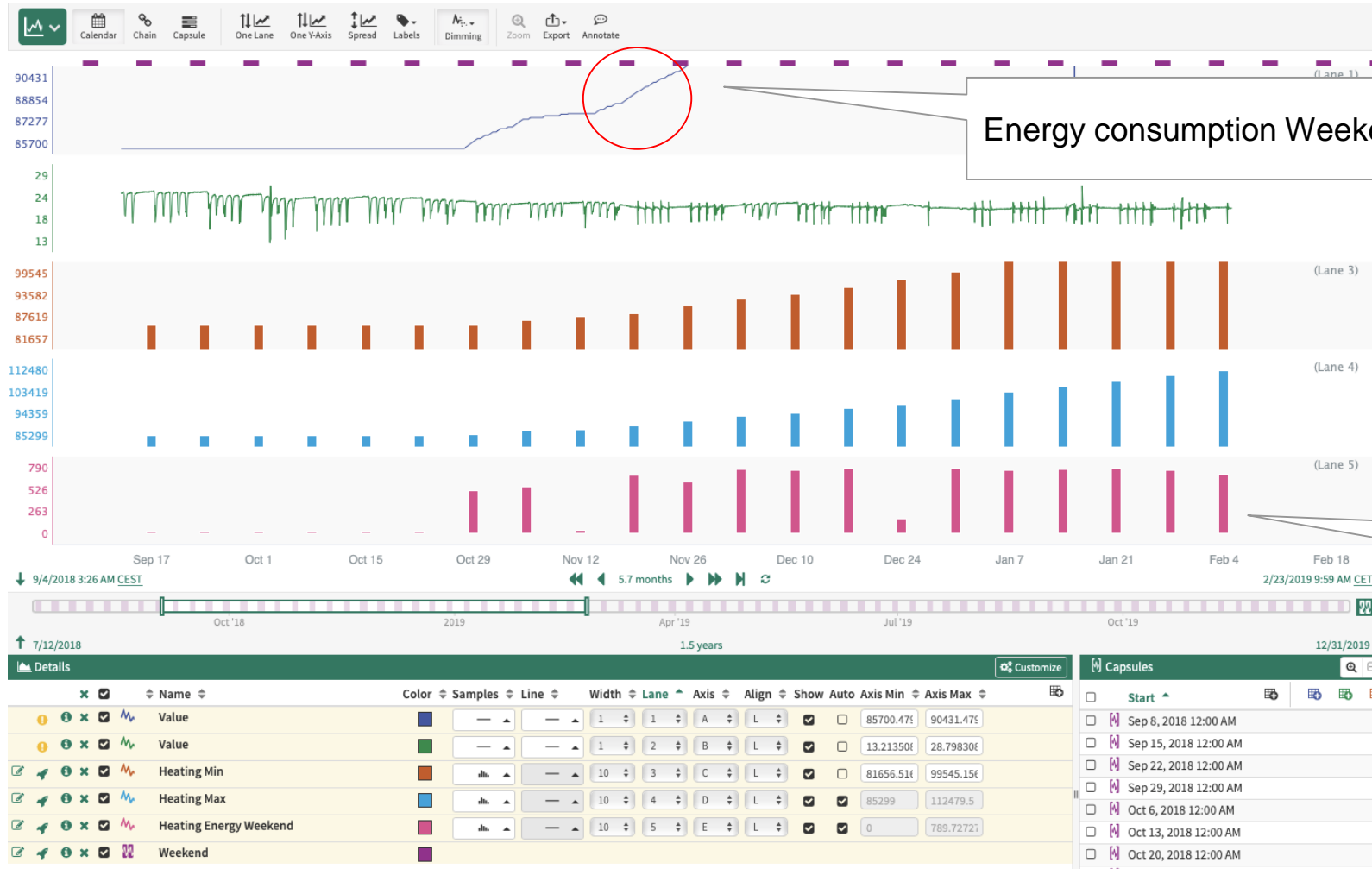


Potential Savings:
3'500 CHF / Year

Grab the attention of your Finance Dept.

Business Case 2

Learning from the next Data



Total energy outside operation

Energy consumption Weekends

Potential Savings:

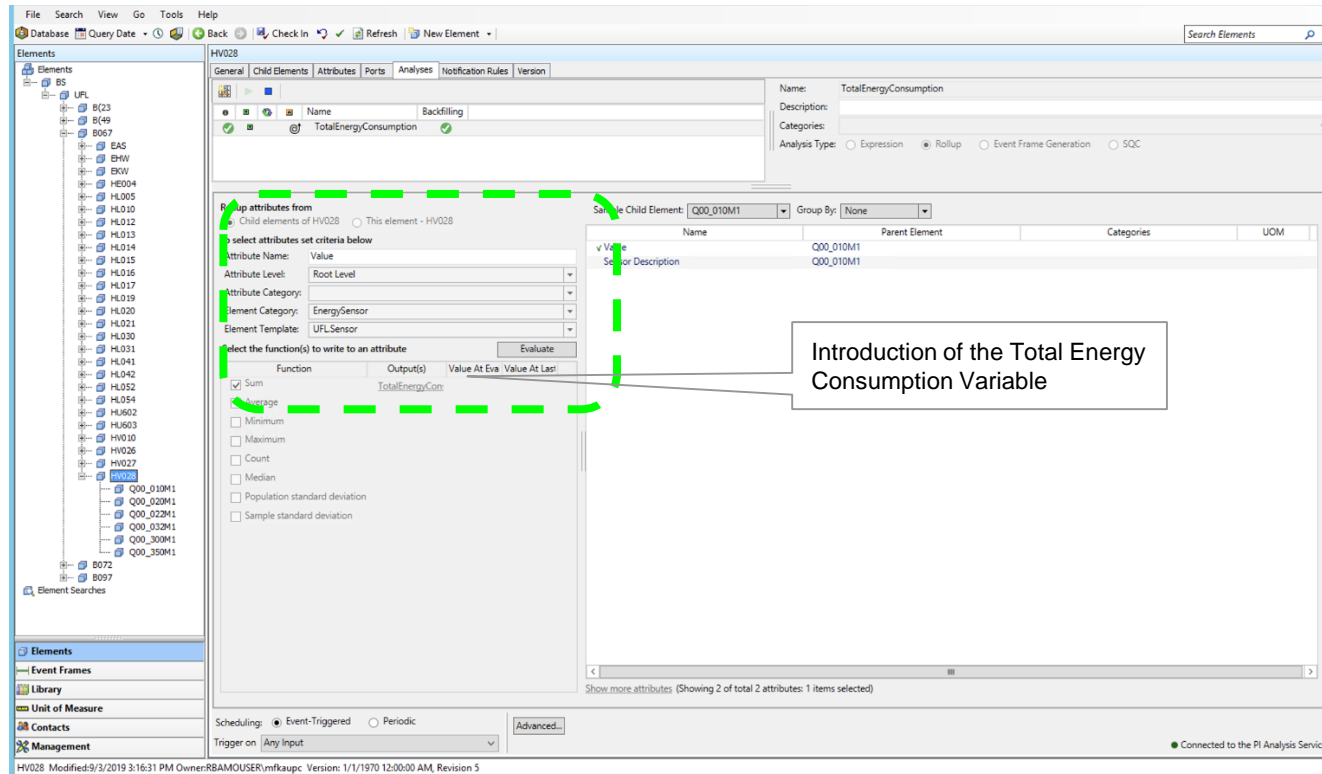
15'560 CHF/Year

Grab the attention of your Finance Dept.

Scaling up from Sensor to building - Data Consolidation in AF

The magic at the core

In order to have the Hypothesis running over all energy tags of the building we consolidate the data in AF:



The screenshot displays the AF software interface for configuring a variable. The left pane shows a tree structure of elements, including UFL, B23, B49, B007, EAS, BHW, BOW, HE004, HLO05, HLO10, HLO12, HLO13, HLO14, HLO15, HLO16, HLO17, HLO19, HLO20, HLO21, HLO30, HLO31, HLO41, HLO42, HLO52, HLO54, HLO60, HLO62, HLO63, HVO10, HVO26, HVO27, HVO28, Q00_010M1, Q00_020M1, Q00_022M1, Q00_032M1, Q00_300M1, Q00_350M1, B072, and B097. The main pane shows the configuration for the 'TotalEnergyConsumption' variable. The 'General' tab is active, showing the variable name, description, and categories. The 'Attributes' tab is also visible, showing the variable's attributes. A red dashed box highlights the 'Sum' function selected in the 'Function' list, with a callout box stating 'Introduction of the Total Energy Consumption Variable'. The 'Output(s)' field shows 'TotalEnergyConsumption'. The 'Scheduling' section is set to 'Event-Triggered' with a trigger on 'Any Input'.

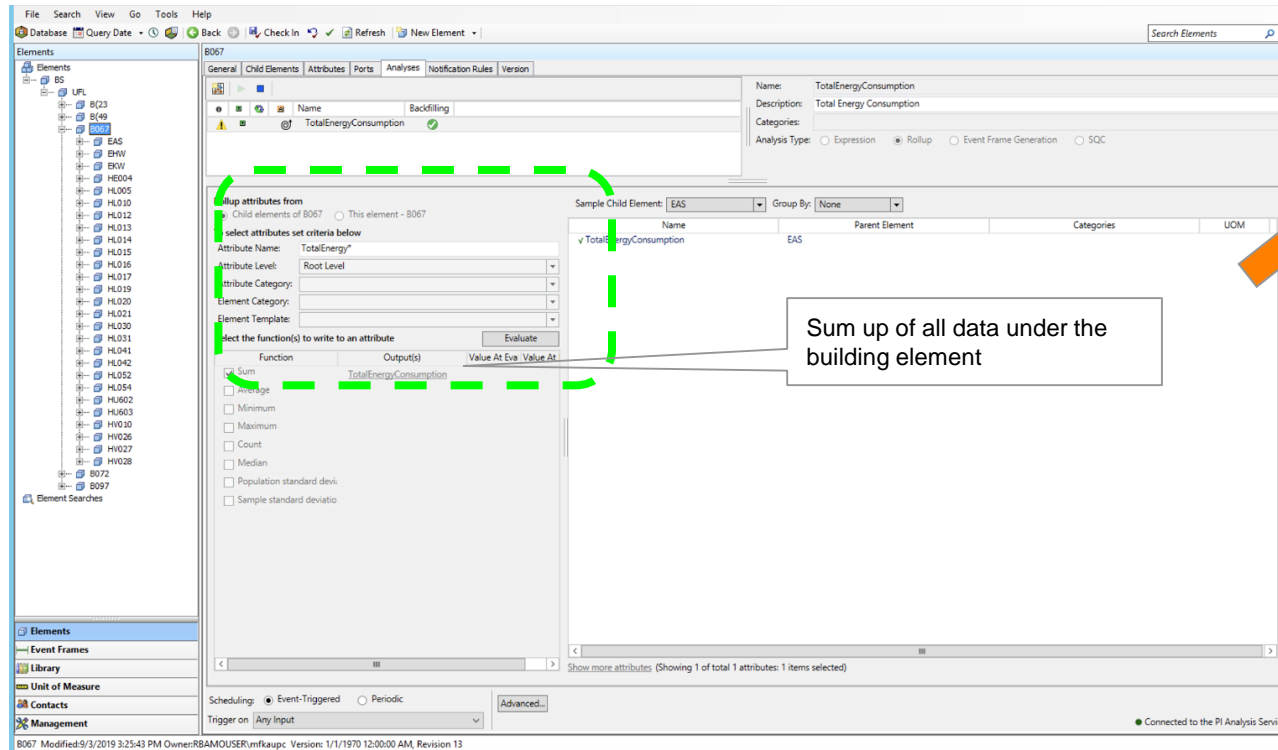
Key elements in the interface include:

- Elements List:** A tree view on the left showing the hierarchy of elements, including UFL, B23, B49, B007, EAS, BHW, BOW, HE004, HLO05, HLO10, HLO12, HLO13, HLO14, HLO15, HLO16, HLO17, HLO19, HLO20, HLO21, HLO30, HLO31, HLO41, HLO42, HLO52, HLO54, HLO60, HLO62, HLO63, HVO10, HVO26, HVO27, HVO28, Q00_010M1, Q00_020M1, Q00_022M1, Q00_032M1, Q00_300M1, Q00_350M1, B072, and B097.
- Configuration Panel:** The main area for configuring the variable, showing tabs for General, Child Elements, Attributes, Ports, Analyses, Notification Rules, and Version.
- Variable Name:** 'TotalEnergyConsumption'.
- Function Selection:** A list of functions including Sum, Average, Minimum, Maximum, Count, Median, Population standard deviation, and Sample standard deviation. The 'Sum' function is selected.
- Output(s):** 'TotalEnergyConsumption'.
- Scheduling:** Set to 'Event-Triggered' with a trigger on 'Any Input'.
- Callout Box:** A red dashed box highlights the 'Sum' function, with a callout box stating 'Introduction of the Total Energy Consumption Variable'.

Data Consolidation in AF

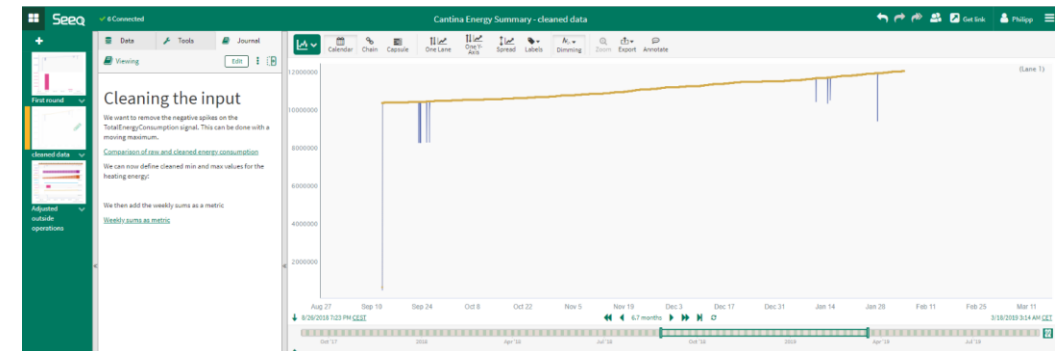
The magic at the core

After introducing the new variable we can sum it up to reflect the energy consumption of the complete building:



Total sum of all energy counters available in PI

Data cleaning and filtering
done with Seeq



End result:
Total Energy Consumption (cleaned) per building

What's next?

The «automated» search for revenue

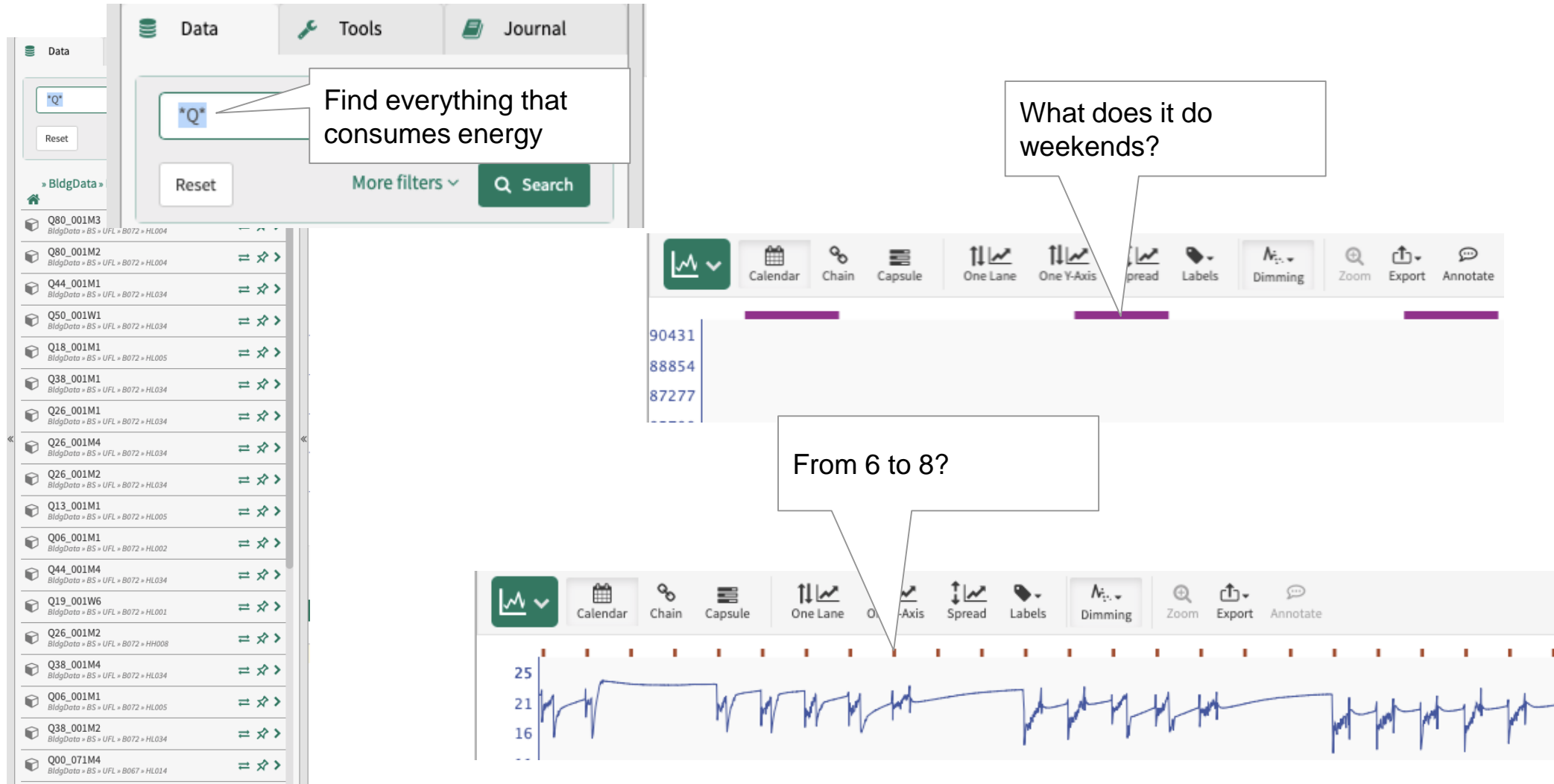
- Automation of further areas of revenue through advanced analytics /machine learning
- Realisation of enhanced control-circuits throughout various source systems
- Long run data storage
- Exchange with future systems
- Connecting the Facility live

DEMO

Life Demo Seeq

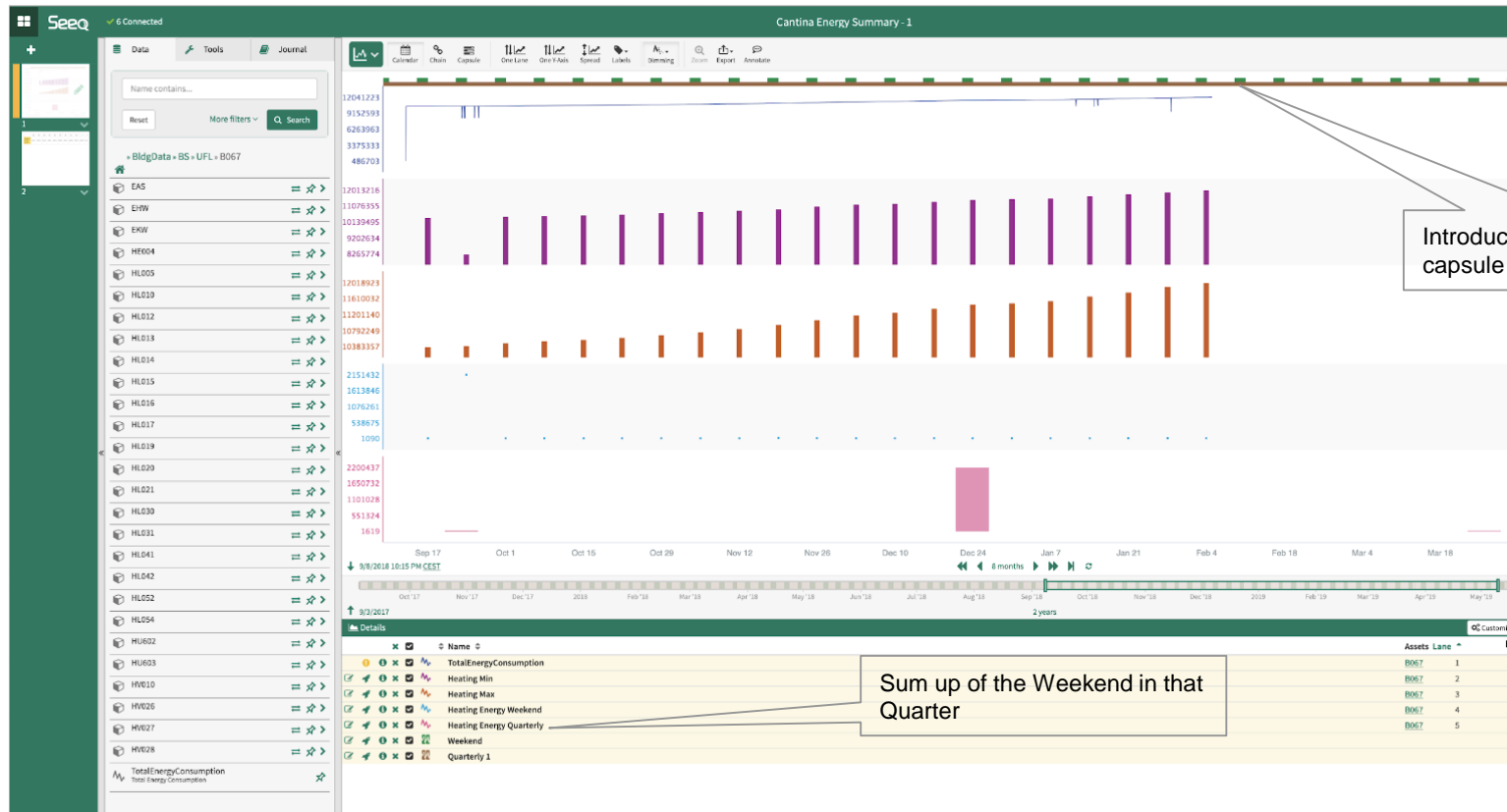
Business Case 2

Automate the search for Revenue



Getting it together in Seeq

After we consolidated the energy data of the complete building, let's view it in a time capsule:



Potential Savings:
24'000 CHF / Year

Potential for adjusted operating times not complete included

Business Case 2

Automate the processes

With the new insight to data it is now time to create dynamic circuits:

- Do a when b etc. (and this over multiple independent systems)
- Forecast outside temperature
- Forecast usage
- Check calendars for usage

As mentioned before, to learn from your data, you can use raw data or any visualisation of your choice. All starts with a simple hypothesis. If the proof of this hypothesis was successful, feel free to scale up and go big!

Challenges for an IoT Plattform

Assumed hurdles

- Data collection from isolated systems
- Data Integrity / Data Ownership
- IT Operations / IT Ownership of a IoT Plattform
- Organisational barriers between contributing departments
- Change in sovereignty of business expertise
- Loss of control over systems due to automated controls
- The investing department may not be benefiting from the savings directly

*Doing now what patients need
next*

Speakers Contact information



- Thorsten Ulbricht
- Product Manager Infrastructure
- F. Hoffmann-La Roche AG
- thorsten.ulbricht@roche.com



- Philipp Sutter
- Dipl. Ing. ETH/SIA, Owner
- OctaveSoft GmbH
- philipp.sutter@octavesoft.ch

Questions?

Please wait for
the **microphone**

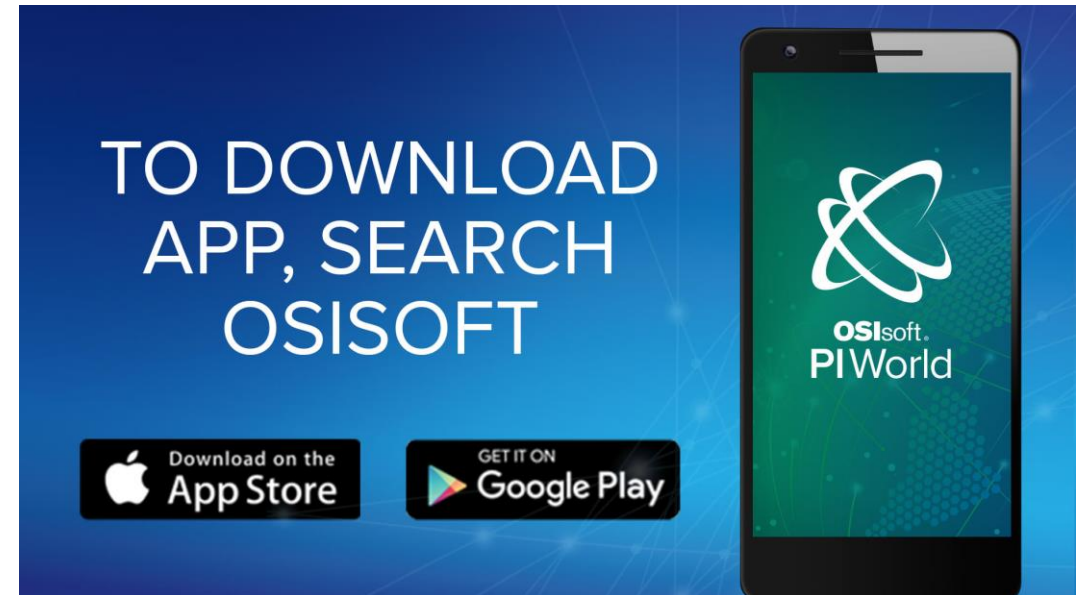
State your
name & company



Please remember to...

Complete Survey!

Navigate to this session in
mobile agenda for survey





PI World

THANK YOU