Software Interlocks System
"helps to protect the machine"

1) Abstract

In the year 2006, a first operational version of a **new Java-based Software Interlock System (SIS)** was introduced to protect parts of the SPS (Super Proton Synchrotron) complex, mainly CNGS (CERN Neutrinos to Gran Sasso), TI8 (SPS transfer line), and for some areas of the SPS ring. SIS protects the machine through surveillance and by analyzing the state of various key devices and dumping or inhibiting the beam if a potentially dangerous situation occurs. Being a part of the machine protection, it shall gradually replace the old SPS Software Interlock System (SSIS) and reach the final operational state targeting LHC (Large Hadron Collider) in Q4 2007. **The system, which was designed with the use of modern, state-of-the-art technologies, proved to be highly successful and very reliable from the very beginning of its existence**. Its relatively simple and very open architecture allows for fast and easy configuration and extension to meet the demanding requirements of the **forthcoming LHC era**.

2) High Energy Beams

Airbus 380:

The energy of an A380 at 700 km/hour corresponds to the energy stored in the LHC magnet system.

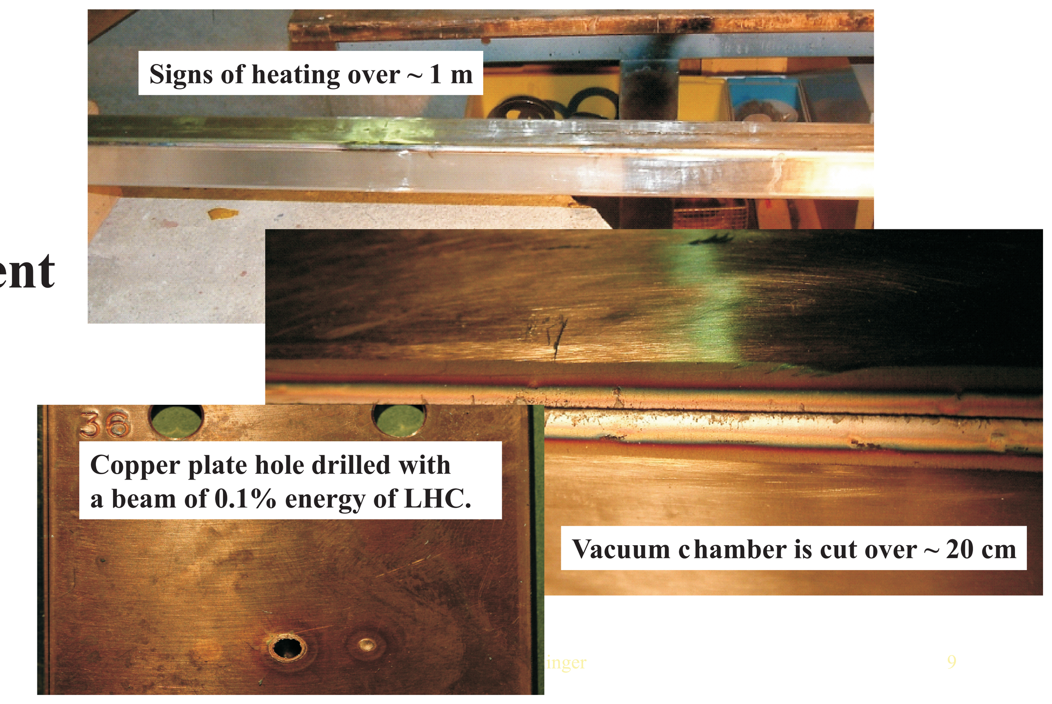
The energy stored in an SPS and LHC beam is orders of magnitude above the damage level of accelerator components like vacuum chambers, magnets, etc.

Energy stored in beam

The energy of one shot (5 kg) at 43'000 km/hour corresponds to the 350 MJ of energy stored in the beams

High energy beams are dangerous for equipment!

High intensity LHC extraction test in SPS incident in 2004.



3) Beam Interlocks System (Hardware)

Both SPS and LHC must be protected by **Beam Interlocks System**. The role of **BIS** is to prevent injection and extraction or dump the beam whenever a failure may lead to a damage of accelerator components. The BIS reaction times are on the time scale of microseconds, limited mostly by transmission delays in electrical components. BIS is designed to provide very high safety and availability since it has to protect very costly equipment. In practice the coverage is limited for historical and practical reasons. This is the origin of the need for Software Interlocks System to provide further protection of the accelerator.

4) Key Requirements

The SIS system has to:

- **anticipate failures and give early alarms**
- **accommodate complex interlock logic**
- **be flexible**
- **be easy to configure**
- **complement BIS**

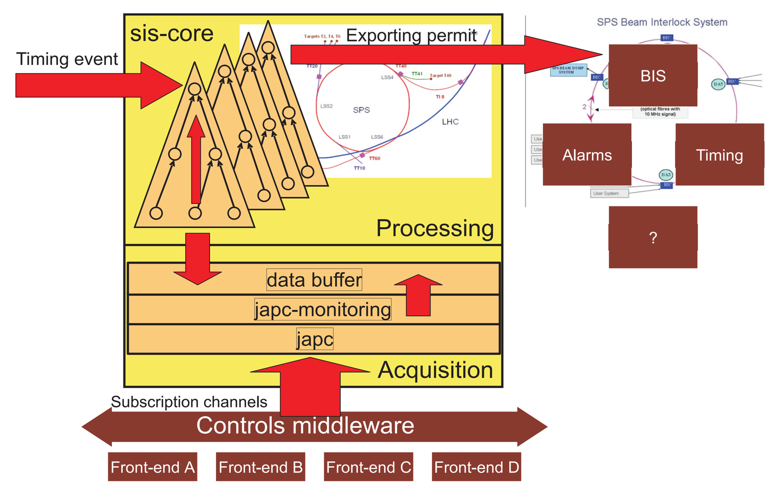
5) Architecture

Layered architecture

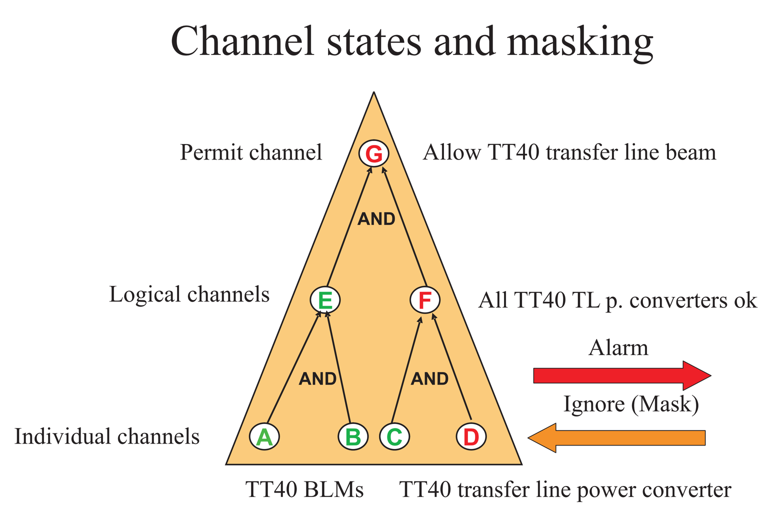
SIS is composed of two main layers working independently from each other:

- **Data Acquisition Layer**
- **Data Processing Layer**

Components are managed by a **Spring container.**



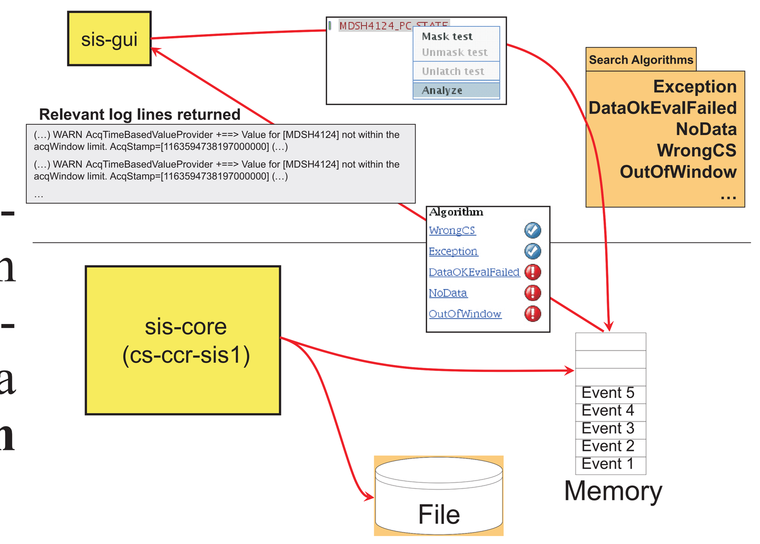
Boolean Expression Permit Tree



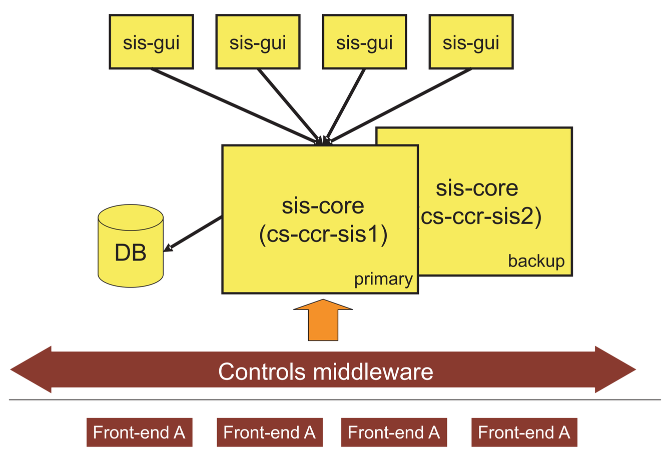
It represents a complex interlock condition on a given set of devices. Leaves are direct **individual channels** that are connected directly to the equipment. Intermediate nodes represents **partial results** (like combined power converters state). The top node is a **Permit signal** that gets exported to external systems (like BIS).

Fault analysis

System operators have an integrated fault analysis tool which searches for typical failure scenarios. It helps answering a question of **why there was an interlock.**



Deployment



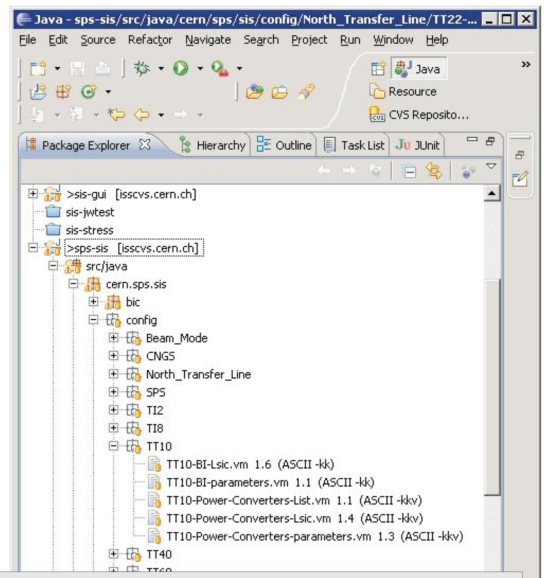
The system is deployed on two machines - a primary one and a backup. It uses an **Oracle database** for persistence and a **JMS broker** for remote access.

6) Configuration

The configuration is stored in a separate project **in CVS**. There are two main XML files:

- **Parameter subscriptions definition**
- **Permit trees definition**

Both are generated with **Velocity template engine** to avoid text duplication and facilitate splitting large files into smaller ones.




```
<Parameters>
 #parse("cern/sps/sis/config/sps-sis-macros.vm")
 <ParameterSet name="CMW-SET" restartPolicy="ONCE">
  (...)
        #parse("cern/sps/sis/config/SPS/LSS4-Septa-
parameters.vm")
        #parse("cern/sps/sis/config/SPS/LSS6-Septa-
parameters.vm")
        #parse("cern/sps/sis/config/SPS/SPS-BIC-
parameters.vm")
  (...)
      #set ( $cnt = 0)
      #foreach( $pcName in $TI2PCList)
            #spsParam($cnt, $pcName)
            #set ($cnt = $cnt +1)
      #end
  (...)
  </ParameterSet>
  <Parameter id="ctim://ctim/CTIM.SPS/SX.SCY-CT" >
     <Module id="SIS"/>
     <Selector id="SPS.USER.ALL" onChange="false"/>
  </Parameter>
  <Parameter id="MDAH2201" name="MDAH2201/STATUS">
        <Module id="SIS"/>
         <Selector id="SPS.USER.ALL" onChange="false"/>
  </Parameter>
  <Parameter id="MDLV2201M"
name="MDLV2201M/STATUS" >
        <Module id="SIS"/>
          <Selector id="SPS.USER.ALL" onChange="false"/>
  </Parameter>
  <Parameter id="QSLD2201" name="QSLD2201/STATUS">
```

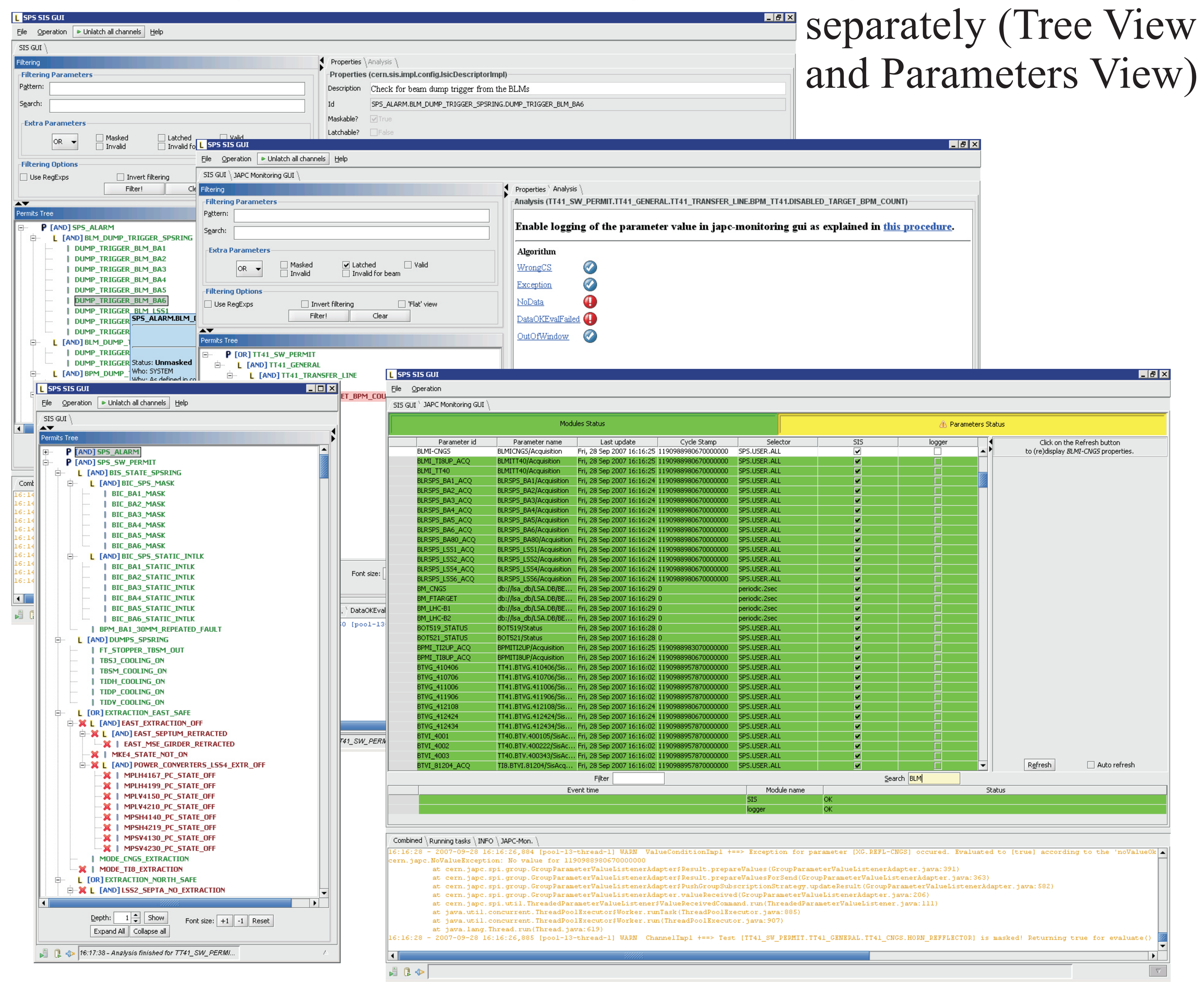
```
<sis:SisConfig
xmlns:sis="http://cern.ch/sis">
  <PermitStructure>

#parse("cern/sps/sis/config/Beam_
Mode/mode-channels.vm")

#parse("cern/sps/sis/config/North_
Transfer_Line/Target-Area-
Lsic.vm")
        (...)
        <Isic id="NORTH_MSE_GIRDER_DOWNSTREAM_RETRACTED"
              latchable="false" masked="false">
              <ValueCondition cycleAware="false" acqWindow="45000"
                    noValueOk="false" parameterId="MST_MSE_NORTH_GIRDER" field="ActPos"
                    index="3" operator="&gt;" value="122" />
        </Isic>
        (...)
        <Lsic id="NORTH_SEPTA_EXTRACTION" latchable="false" masked="false">
              <LogicalCondition operator="AND">
                    <Test refid="NORTH_ZS_HV_ON" />
                    <Test refid="NORTH_ZS1_IONTRAP" />
                     (...)
              </LogicalCondition>
        </Lsic>
        (...)
         <Permit id="TT60_SW_PERMIT" latchable="false">
              <LogicalCondition operator="OR">
                    <Test refid="MODE_TT60_NO_EXTRACTION" />
                    <Test refid="LSS6_TT60_TRANSFER_LINE" />
               </LogicalCondition>
              <Exporters>
                    <BICExporter target="CIB.BA6.TT60A" />
                    <TimingExporter target="I_S.SIS_TT60"/>
               <AlarmExporter/>
              </Exporters>
```


7) Graphical User Interface

The core of the system is controlled by a **GUI interface**. Its main goal is to provide a view to the **configuration** and **current state** of the system. Operators can **see the faulty channels** and **analyze** them with the analysis tool. Two main layers of the system are viewed separately (Tree View and Parameters View)



8) Conclusions

As of today SIS successfully surveys over **1200 individual software channels for the SPS accelerator.** It has become a **vital tool** for everyday operations. It will be used to **control the LHC machine**. The applied architecture has proven itself **very reliable** in practice making the SIS project a **great success at CERN**.

Jakub Wozniak

V. Baggiolini, J. Wenninger, D.G. Quintas