SMIng Project and Related Work

Jürgen Schönwälder

Computer Science Department
Technical University Braunschweig
Bültenweg 74/75
38106 Braunschweig
Germany

Phone: +49 531 391 3289
Email: `schoenw@ibr.cs.tu-bs.de`
Web: `http://www.ibr.cs.tu-bs.de/~schoenw/`

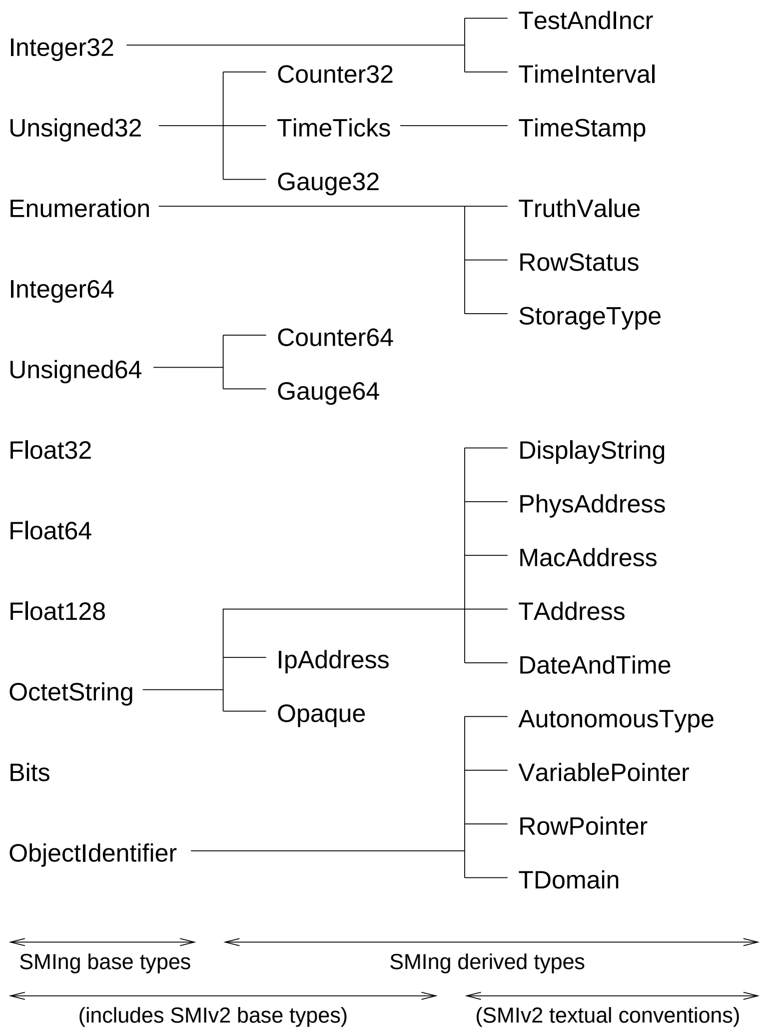
Original SMIng Project Goals (Fall 1998)

- Design a next generation SMI that is "easier to use."
- SMIng must be compatible with SMIv2 so that automatic translations are possible.
- Work upwards from the SMIv2 towards a richer data definition language.
- SMIng language definition must be self-contained and not rely on external standards.
- SMIng must support a complete set of base data types.
- Simplify the SMI language wherever possible.
  - SMIng should use a small set of simple and consistent syntactic rules.
  - SMIng must be programmer friendly and easy to read/write for humans.
  - SMIng should be compact and must avoid redundant definitions.
  - SMIng should disallow forward references except in cases where they are needed.
  - Avoid to overspecify language constructs.
- SMIng must model table existance relationships.
- SMIng must be extensible so that enhancements can be introduced gradually.

SMIng Syntax Rules

- An SMIng file simply contains a sequence of SMIng statements.
- Every SMIng statement starts with a leading lowercase keyword identifying the statement followed by a number of arguments.
- An argument may be
  - quoted text or
  - an identifier or
  - a value of any base type or
  - a type restriction or
  - a list of identifiers enclosed in parenthesis or
  - a statement block enclosed in curly braces.
- Statements are terminated by a semicolon.
- It is possible to nest statement sequences since statement blocks are valid arguments.
- Uppercase identifiers identify SMIng types and modules.
- Lowercase identifiers identify statements, named numbers or nodes in the OID registration tree.
- Comments start with the character sequence // and end at the end of the line.

SMIng Base Types and Core Derived Types



- SMIng allows to derive a new types from a derived types.
- Used to defined some of the SMIv2 base types as dervived types.
- Type definitions can have "units" statements.

IF-MIB Module in SMIng

```
module IF-MIB ifMIB {

  import IRTF-NMRG-SMING-TYPES (Counter32, Gauge32, Counter64, TimeTicks
                               DisplayString, PhysAddress, TruthValue,
                               RowStatus, TimeStamp, AutonomousType,
                               TestAndIncr);
  import IRTF-NMRG-SMING (mib-2);
  import SNMPv2-MIB (snmpTraps);
  import IANAifType-MIB (IANAifType);

  oid          mib-2.31;

  organization  "IETF Interfaces MIB Working Group";
  contact       "   Keith McCloghrie
                    Cisco Systems, Inc.
                    170 West Tasman Drive
                    San Jose, CA 95134-1706
                    US

                    408-526-5260
                    kzm@cisco.com";
```

```
  description   "The MIB module to describe generic objects for
                 network interface sub-layers.  This MIB is an updated
                 version of MIB-II's ifTable, and incorporates the
                 extensions defined in RFC 1229.";

  revision {
    date        "1996-11-03 13:55";
    description "[Revision added by libsmi due to a LAST-UPDATED clause.]";
  };
  revision {
    date        "1996-02-28 21:55";
    description "Revisions made by the Interfaces MIB WG.";
  };
  revision {
    date        "1993-11-08 21:55";
    description "Initial revision, published as part of RFC 1573.";
  };

  // extension statements
  // typedef statements
  // node/scalar/table statements
  // notification statements
  // group statements
  // compliance statements
};
```

Definition of InterfaceIndex in SMIng

```
typedef InterfaceIndex {
  type        Integer32 (1..2147483647);
  format      "d";
  description "A unique value, greater than zero, for each interface
               or interface sub-layer in the managed system.  It is
               recommended that values are assigned contiguously
               starting from 1.  The value for each interface sub-
               layer must remain constant at least from one re-
               initialization of the entity's network management
               system to the next re-initialization.";
};
```

- The status of a definition is "current" if the status statement is absent.
- SMIng does similar to the SMIv2 not support constructed types.

Definition of ifNumber in SMIng

```
scalar ifNumber {
  oid         interfaces.1;
  type        Integer32;
  access      readonly;
  description "The number of network interfaces (regardless of their
               current state) present on this system.";
};
```

- SMIng makes the distinction between scalars, tables, rows and columns explicit.

Definition of ifTable in SMIng

```
table ifTable {
  oid         interfaces.2;
  description "A list of interface entries.  The number of entries
               is given by the value of ifNumber.";

  row ifEntry {
    oid         ifTable.1;
    index       (ifIndex);
    description "An entry containing management information applicable
                 to a particular interface.";

    column ifIndex {
      oid         ifEntry.1;
      type        InterfaceIndex;
      access      readonly;
      description "A unique value, greater than zero, for each
                   interface.  [...]";
    };
    // ...
  };
};
```

Extending the SMIng to Identify Discontinuity Indicators

```
module TUBS-IBR-EXTENSIONS {

  //...

  extension discontinuity {
    description "Indicates the discontinuity indicating object for a
                 given set of targets. The intended usage of this
                 extension is within an annotation statement which
                 identifies the targets. The ABNF definition for this
                 extension is:

                   discontinuityStatement = discontinuityKeyword sep
                                            qlcIdentifier optsep ';'
                   discontinuityKeyword = 'discontinuity'
                 ";
  };
};
```

Using the SMIng Language Extension in an Annotation

```
module TUBS-IBR-IF-MIB-ANNOTATION {

  imports IRTF-NMRG-SMING-EXTENSIONS (annotation);
  imports TUBS-IBR-EXTENSIONS (discontinuity);

  //...

  annotation ifTableDiscontinuity {
    targets       (ifInOctets, ifInUcastPkts, ifInNUcastPkts,
                   ifInDiscards, ifInErrors, ifInUnknownProtos,
                   ifOutOctets, ifOutUcastPkts, ifOutNUcastPkts,
                   ifOutDiscards, ifOutErrors);
    discontinuity ifCounterDiscontinuityTime;
    description   "The discontinuity indicator for the ifTable.";
  };
};
```

Implementation

- Proof of concept implementation available as part of the `libsmi` package.

- `libsmi` API hides 99% of the SMI version differences from the application.

- Translations implemented in the `smidump` tool contained in the `libsmi` package:
  - SMIng → SMIv2 (for the subset where translations is possible)
  - SMIv2 → SMIng
  - SMIng → CORBA IDL (consistent with JIDM specifications)
  - SMIng → MOSY file format (for the subset possible)
  - SMIng → XML DTD (according to <draft-irtf-nmrg-smi-xml-00.txt>)
  - SMIv2/SMIng → UML class diagrams

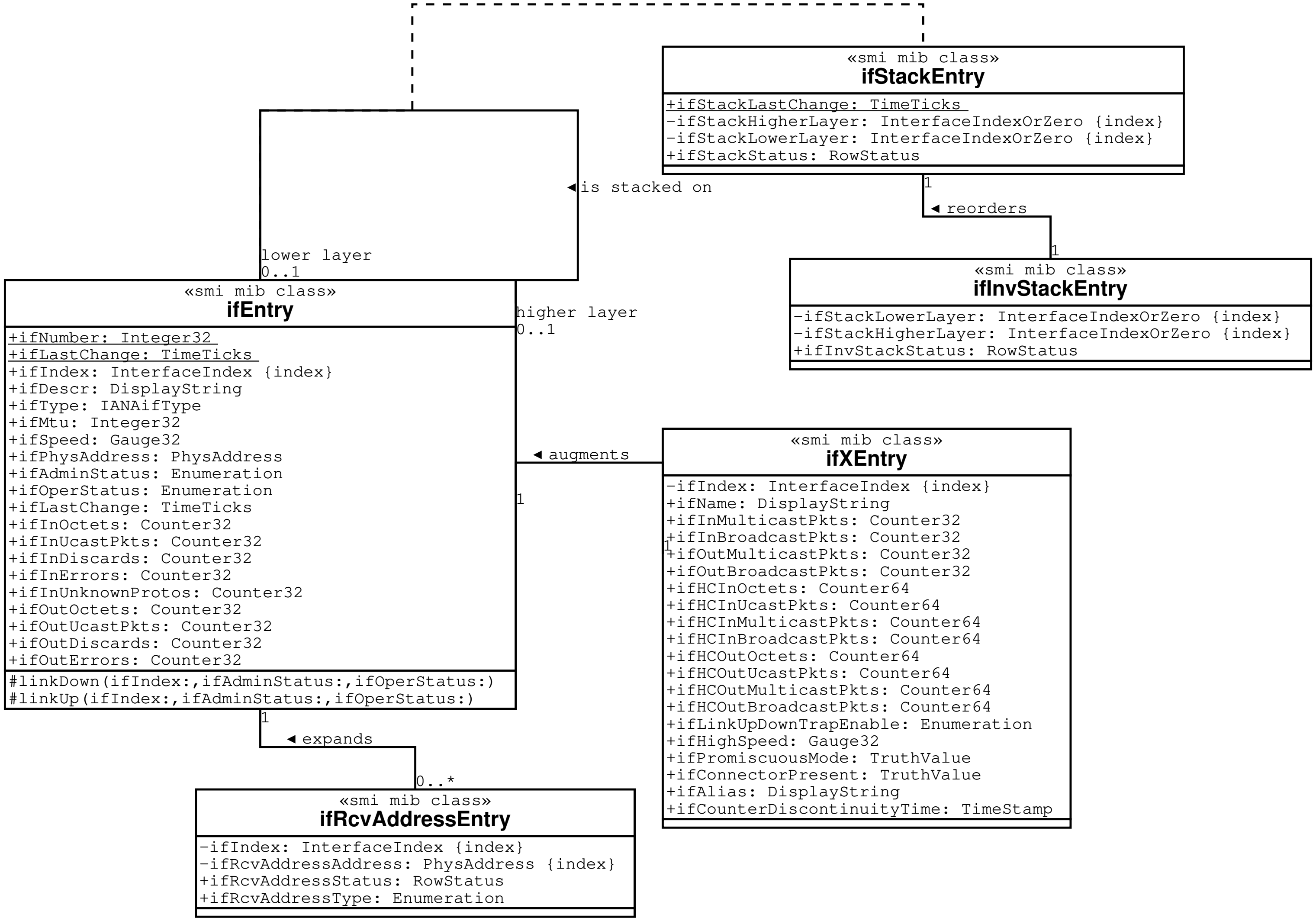
- Freely available from <URL:http://www.ibr.cs.tu-bs.de/projects/libsmi/>.

- Integrated into several open source and a number of commercial packages.

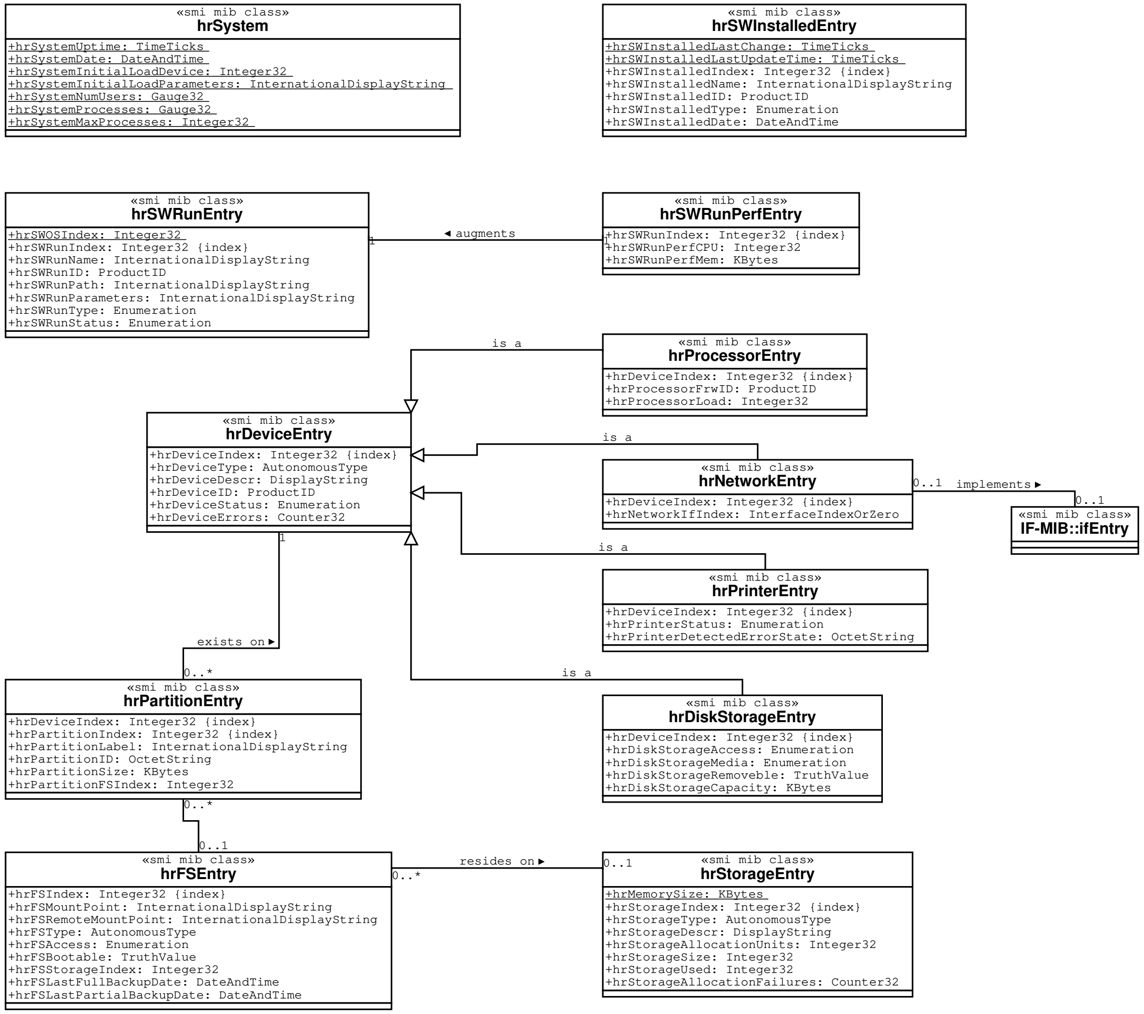
Reverse Engineering MIBs

- Most MIBs are based on "conceptual model."
- Many MIB authors to not bother to document this "conceptual model."
- An explicit "conceptual model" reduces the time needed to understand a MIB module.
- Information about the "conceptual model" can greatly improve the effectiveness of code generators.
- The Universal Modeling Language (UML) has become the lingua franca in software engineering.
- Many heuristics are needed to reverse engineer SMIv2 MIBs into UML diagrams.
- SMIng specifications can much easier be mapped into UML diagramms since SMIng makes much more modeling information available in a machine readable format.

IF-MIB and IF-INVERTED-STACK-MIB



HOST-RESOURCES-MIB



Summary

- SMIng is an important evolutionary step for the SMIv2.
- I seems to be possible to join SMIv2 and SPPI in the SMIng.
- SMIng's extensibility features allow to enhance the data definition language over time.
- Being able to feed model/data definitions into computer programs is a key requirement.
- It must be for humans to read/edit model/data definitions.
- It is required to define algorithmic mappings to exiting technologies.
- Working bottom-up from where things are today is very effective to get algorithmic mappings working.
- Working top-down has a high risk to fail since algorithmic mappings often turn out to be too hard when you start with too many features.

More Information

- SMIng specification <draft-irtf-nmrg-sming-01.txt>
- SMIng home page <URL:http://www.ibr.cs.tu-bs.de/projects/sming/>
- SMIng implementation <URL:http://www.ibr.cs.tu-bs.de/projects/libsmi/>
- NMRG archive <URL:http://www.ibr.cs.tu-bs.de/projects/nmrg/>
- J. Schönwälder, F. Strauß. Next Generation Structure of Management Information for the Internet, 10th IFIP/IEEE Workshop on Distributed Systems: Operations and Management", "Springer Verlag", October 1999.
- J. Schönwälder, A. Müller. Reverse Engineering Internet MIBs (in preparation for publication)