Appendix C

UML Benchmark Results

This appendix presents a number of figures that complete the graphs presented during the UML analysis in chapter 6. These were not included in the main body of the thesis because many results share the same features. Also, they may only differ in time scales due to the performance of each test platform.

The charts in Figure C.1 complement those in Figure 6.2.

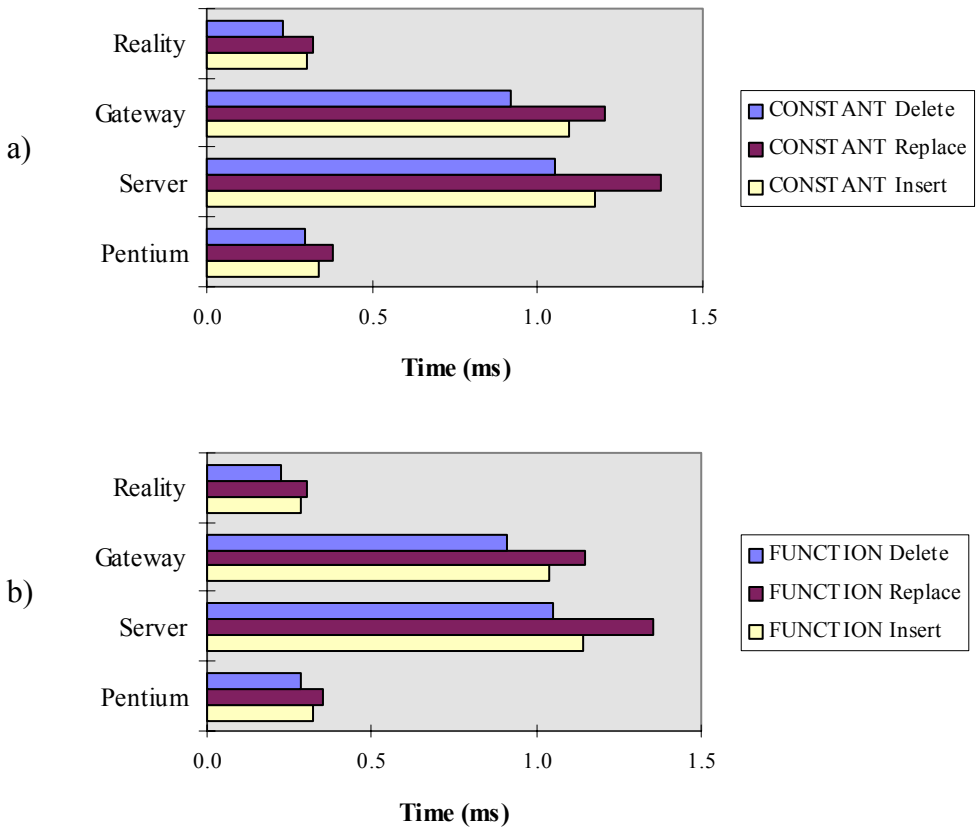
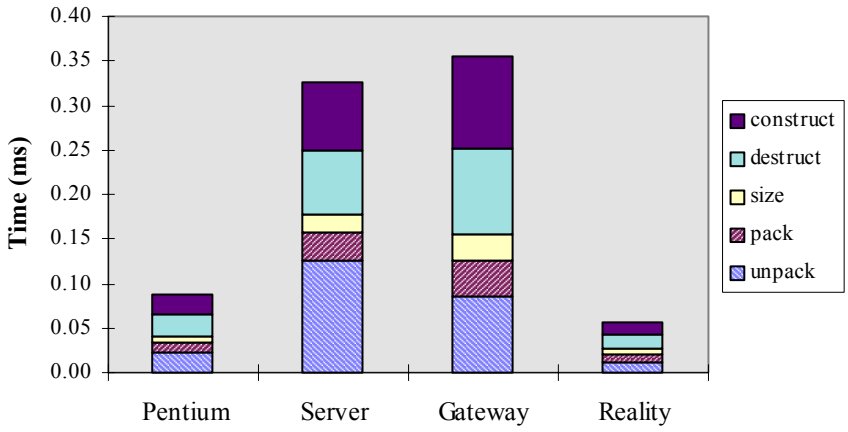


Figure C.1 Basic interpreter overheads for two primitive types: a) Constant; b) Function.

The complementing charts to Figure 6.4 can be found in Figure C.2.

a)



b)

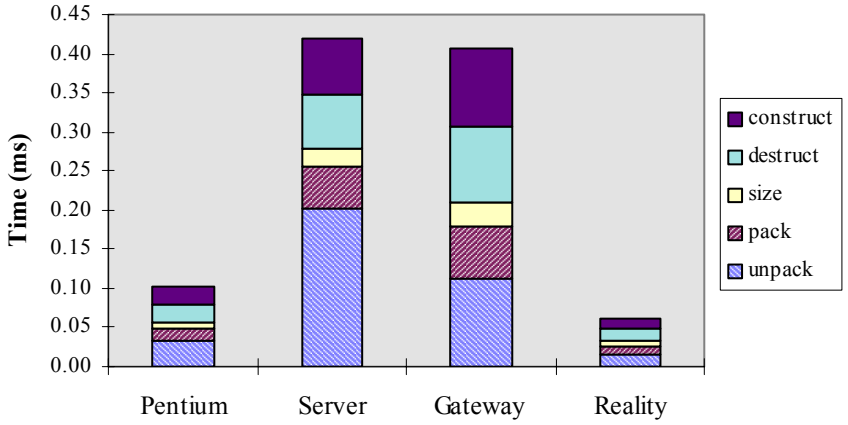
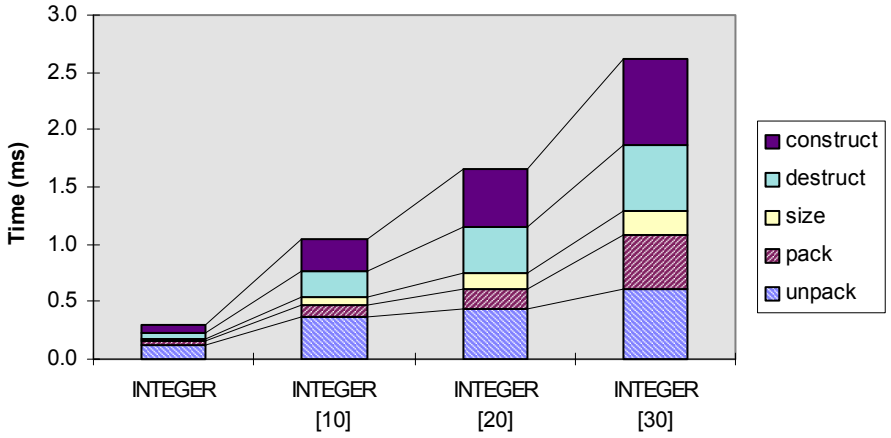


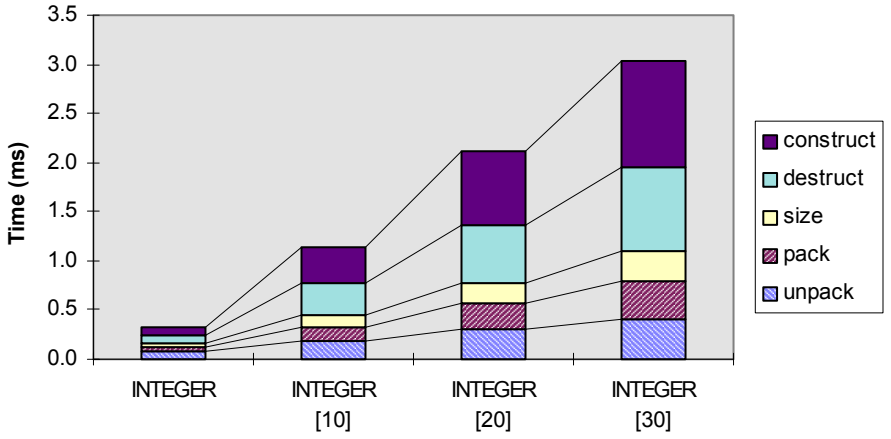
Figure C.2 Cost of fundamental state operations on a String with: a) 0 characters; b) 40 characters.

Figure C.3 complements Figure 6.5 which presented state operation costs for Pentium.

a)



b)



c)

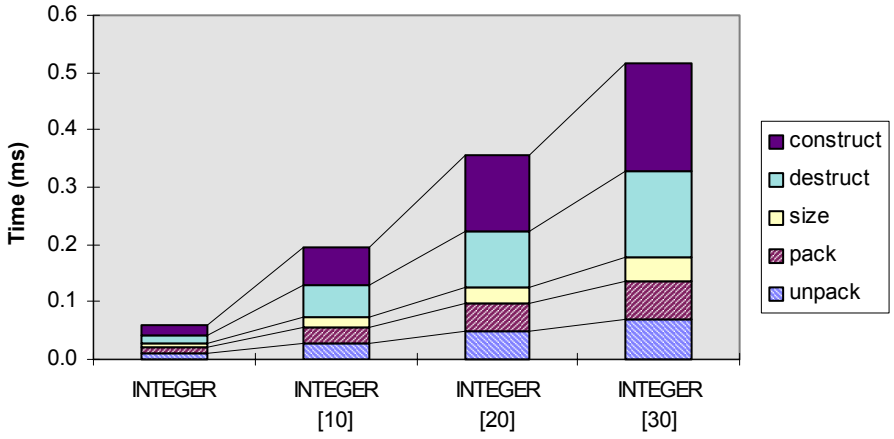


Figure C.3 State operation costs based upon state size: a) Server; b) Gateway; c) Reality.

Figure C.4 shows the state operation overheads for the three platforms not shown in Figure 6.6.

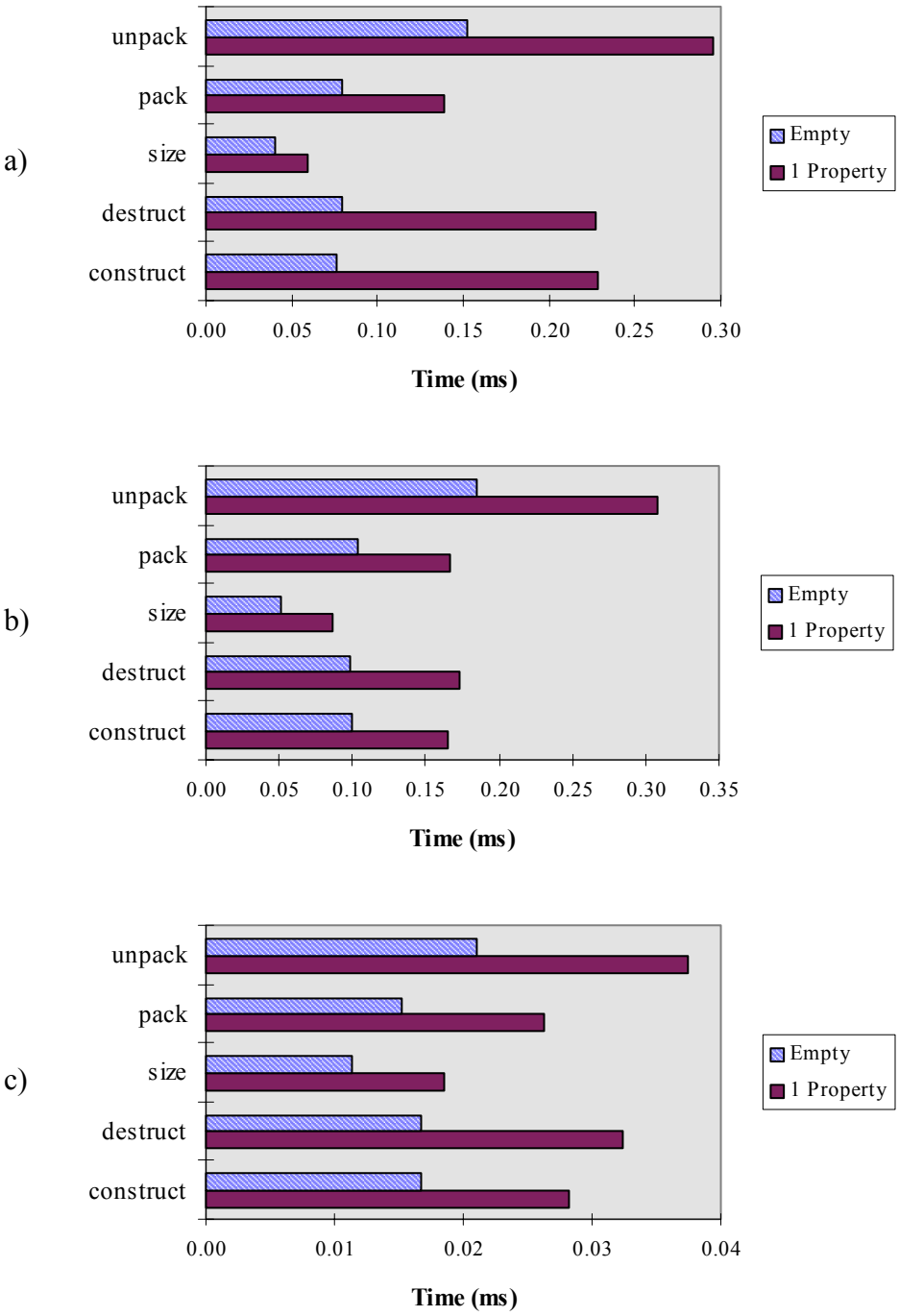


Figure C.4 State operation overheads for an element with zero and one properties: a) Server; b) Gateway; c) Reality.

The complement of Figure 6.7 is shown in Figure C.5.

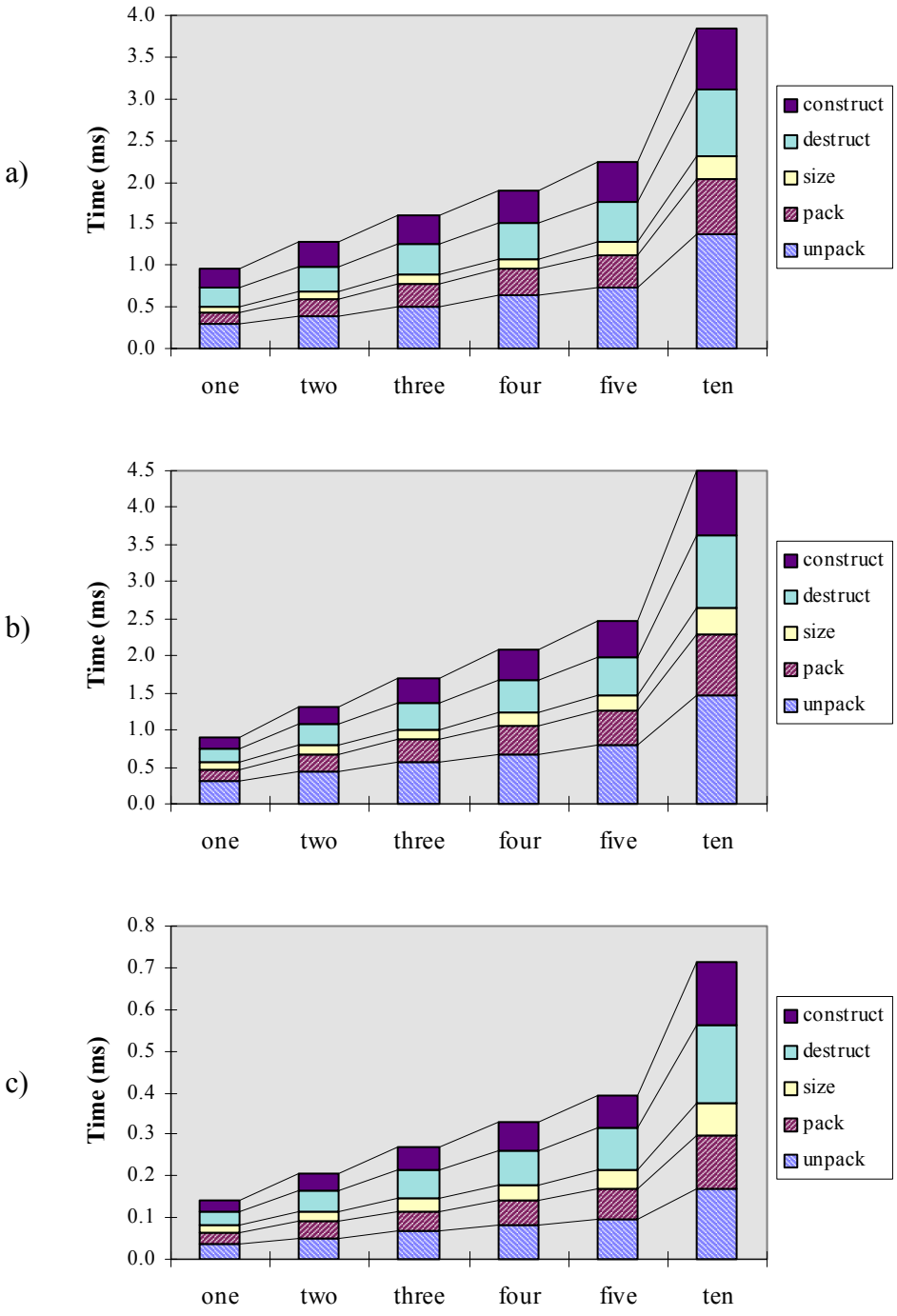
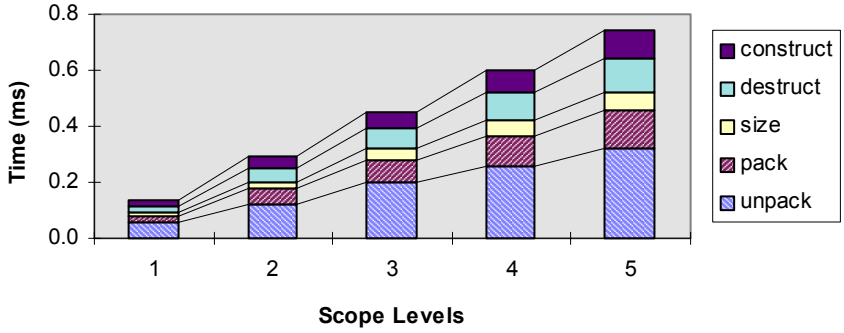


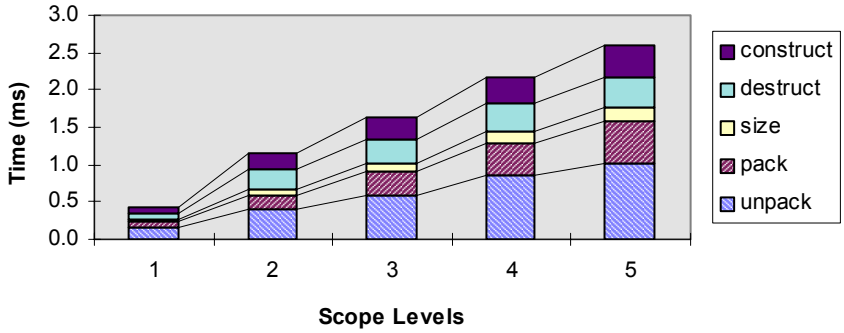
Figure C.5 State operations on elements with one to ten properties: a) Server; b) Gateway; c) Reality.

Figure C.6 shows how different levels of scope (nested elements) affect state operation overheads.

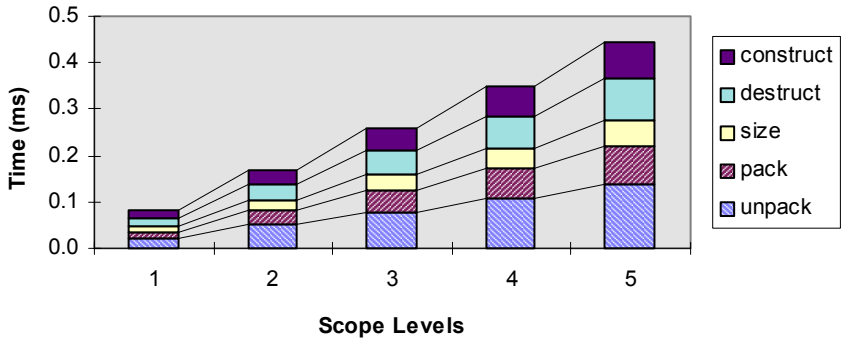
a)



b)



c)



a)

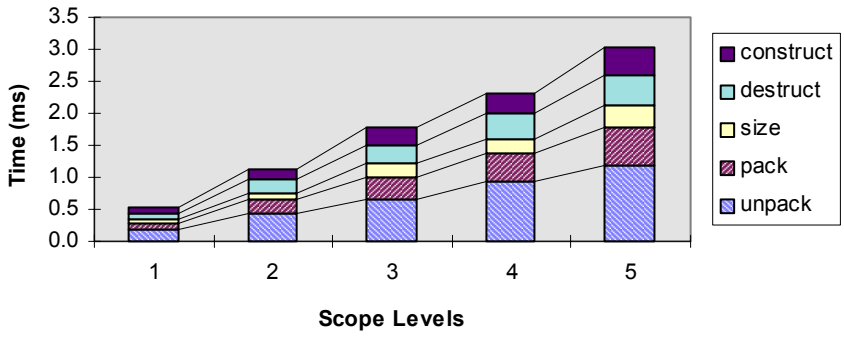


Figure C.6 State operation costs based upon scope level: a) Pentium; b) Server; c) Gateway; d) Reality.

The UML definition used for these tests is given in Figure C.7.

```
UNIVERSE Base
{
    CONSTANT      c      : INTEGER = 1;

    ELEMENT Empty
    {
    }

    ELEMENT One
    {
        PROPERTY  prop1  : INTEGER;
    }

    ELEMENT Two
    {
        PROPERTY  prop1  : INTEGER;
        PROPERTY  prop2  : INTEGER;
    }

    ELEMENT Three
    {
        PROPERTY  prop1  : INTEGER;
        PROPERTY  prop2  : INTEGER;
        PROPERTY  prop3  : INTEGER;
    }

    ELEMENT Four
    {
        PROPERTY  prop1  : INTEGER;
        PROPERTY  prop2  : INTEGER;
        PROPERTY  prop3  : INTEGER;
        PROPERTY  prop4  : INTEGER;
    }

    ELEMENT Five
    {
        PROPERTY  prop1  : INTEGER;
        PROPERTY  prop2  : INTEGER;
        PROPERTY  prop3  : INTEGER;
        PROPERTY  prop4  : INTEGER;
        PROPERTY  prop5  : INTEGER;
    }

    ELEMENT Ten
    {
        PROPERTY  prop1  : INTEGER;
        PROPERTY  prop2  : INTEGER;
        PROPERTY  prop3  : INTEGER;
        PROPERTY  prop4  : INTEGER;
        PROPERTY  prop5  : INTEGER;
        PROPERTY  prop6  : INTEGER;
        PROPERTY  prop7  : INTEGER;
        PROPERTY  prop8  : INTEGER;
        PROPERTY  prop9  : INTEGER;
        PROPERTY  prop10 : INTEGER;
    }
```

```
ELEMENT Elem
{
    PROPERTY  anInteger      : INTEGER;
    PROPERTY  aReal          : REAL;
    PROPERTY  aBoolean       : BOOLEAN;
    PROPERTY  aString        : STRING;
}

ELEMENT Level2
{
    ELEMENT Level1
    {
    }

    PROPERTY  level1 : Level1;
}

ELEMENT Level3
{
    ELEMENT Level2
    {
        ELEMENT Level1
        {
        }

        PROPERTY    level1   : Level1;
    }

    PROPERTY  level2 : Level2;
}

ELEMENT Level4
{
    ELEMENT Level3
    {
        ELEMENT Level2
        {
             ELEMENT Level1
             {
             }

             PROPERTY       level1   : Level1;
        }

        PROPERTY    level2   : Level2;
    }

    PROPERTY  level3 : Level3;
}

// Continued...
```

```
    ELEMENT Level5
    {
        ELEMENT Level4
        {
            ELEMENT Level3
            {
                  ELEMENT Level2
                  {
                      ELEMENT Level1
                      {
                      }

                      PROPERTY   level1 : Level1;
                  }

                  PROPERTY       level2  : Level2;
            }

            PROPERTY    level3  : Level3;
        }

        PROPERTY  level4 : Level4;
    }

    PROPERTY      empty          : Empty;
    PROPERTY      elem           : Elem;
    PROPERTY      one            : One;
    PROPERTY      two            : Two;
    PROPERTY      three          : Three;
    PROPERTY      four           : Four;
    PROPERTY      five           : Five;
    PROPERTY      ten            : Ten;

    PROPERTY      level2         : Level2;
    PROPERTY      level3         : Level3;
    PROPERTY      level4         : Level4;
    PROPERTY      level5         : Level5;

    PROPERTY      intList10      : INTEGER[10];
    PROPERTY      intList20      : INTEGER[20];
    PROPERTY      intList30      : INTEGER[30];
}
```

Figure C.7 UML definition used in the interpreter benchmarks.