```
class Schema
{
    private string pattern = @"^\s*(?<redefines>R?)(?<level>\d+)\s+(?<varName>\S+)((\s+(?<type>[xcnpXCNP])\s+((?<length>\d+))(\,(?<decimalPlaces>\d+))?)?(\s+(?<repeatCount>\d+))?)?(\s{2,}(?<comment>.*))?$";
    private string schemaStr = "";

    public AbstractNode ParseLine(string line)
    {
        // gibt eine Value- oder GroupNode zurück, je nachdem ob line
        // Angaben zu Typ und ByteAnzahl hat oder nicht
        // wenn pattern nicht matcht wird null zurückgegeben
    }

    public GroupNode Parse()
    {
        var stack = new Stack<AbstractNode>();
        var root = new GroupNode(false, 0, "root", 1, 1, "");
        stack.Push(root);

        Action addChildFromStackToParent =
            () =>
            {
                var child = stack.Pop();
                var parent = stack.Peek();
                parent.AddChild(child);
                for (int currentRepeatIndex = 2; currentRepeatIndex <= child.RepeatCount; ++currentRepeatIndex)
                {
                    parent.AddChild(child.CreateCopyWithIndex(currentRepeatIndex));
                }

            };

        var schemaLines = schemaStr.Split(new string[] { "\r\n", "\n" }, StringSplitOptions.RemoveEmptyEntries);
        foreach (var currentLine in schemaLines)
        {
            var currentNode = ParseLine(currentLine);
            if (currentNode == null) { continue; }
            while (currentNode.Level <= stack.Peek().Level)
            {
                addChildFromStackToParent();
            }
            stack.Push(currentNode);
        }
        while (stack.Count >= 2)
        {
            addChildFromStackToParent();
        }
        return root;
    }
}
```