アプリケーションの構造を視覚化する nesC 用開発環境の提案

村上 貴彦† 松尾 英治†† 鈴来 和久†† 横田 裕介† 大久保 英嗣†
†立命館大学情報理工学部 ††立命館大学大学院理工学研究科

1 はじめに

現在, センサノードには TinyOS と呼ばれるセンサノード用 OS が多く採用されている. TinyOS は, 限られた資源を有効利用するために, 独自のコンポーネント指向言語である nesC [1] によって記述されている. 開発者は, TinyOS を構成するコンポーネントを一部書き換えたり追加することによって目的のソフトウェアの開発を行う. しかし, nesC では, コンポーネントの粒度が小さい上, イベント駆動で並列に処理を行うデザインのため, コンポーネント間の関連が複雑化しやすいという問題がある. これはコンポーネント数が多くなるほど大きな問題となり, nesC におけるソフトウェアの開発を難しくする原因となっている. 本稿では, ソフトウェアの構造であるコンポーネント間の関連を視覚化する開発環境を提案する.

2 nesC の概要と課題

2.1 nesC によるプログラミングの概要

nesC は, TinyOS を記述するために考案された C 言語の拡張言語である. 開発者は, nesC で開発を行うことにより, 柔軟性が高く, 限られた資源下で動作するソフトウェアを短期間で開発することが可能である.

nesC では, コンポーネントによってソフトウェアを構成する. コンポーネントは, さらにモジュールとコンフィグレーションに分類される. モジュールは, 処理内容が記述された要素であり, 提供するインタフェースのコマンド, および利用するインタフェースのイベントが実装されなければならない. コンフィグレーションは, ワイアリングと呼ばれるインタフェース同士の接続が記述される要素である. コンフィグレーションの実装例を図 1 に示す. 図中の TimerC がコンフィグレーションであり, TimerM と HWClockM は内包コンポーネントと呼ばれるコンフィグレーションを構築するために利用されるコンポーネントである. TimerM と HWClockM の接続のような, 内包コンポーネント同士の接続をリンクワイアリングと呼ぶ. リンクワイアリングは, 提供インタフェースと利用インタフェースを接続する. また, TimerC と TimerM のような内包コンポーネントとコンフィグレーション間の接続を同等ワイアリングと呼ぶ. 同等ワイアリングでは, 提供インタフェース同士, または利用インタフェース同士で接続が行われる. それぞれのインタフェースでは, 利用側からの命令であるコマンドと, 提供側から事象の発生を通知するイベントが定義されている.

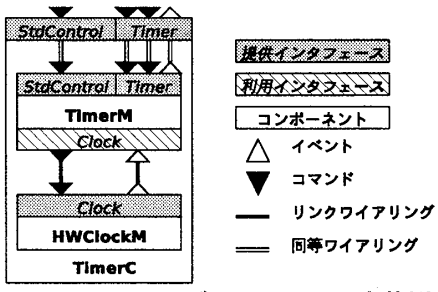

図 1 コンフィグレーションの実装例

2.2 nesC を用いた開発の課題

nesC を用いた場合, 開発者は, 開発を行うための事前調査や開発工程そのものに多くの時間を要する. これは, 記述し直すべき適切な個所を多くのファイルの中から探さなければならず, また, 処理するソースコード間を行き来することに多くの時間を要するからである. これらの問題点は, nesC の次の特徴によって引き起こされる.

- コンポーネント間の関連の複雑さ
  nesC で実装されるコンポーネントは, 様々なハードウェア環境に適応するため, 粒度の細かい実装が行われている. そのため, ソフトウェアは多くのコンポーネントにより構成され, 関連が複雑化しやすい.
- 処理の流れの見通しの悪さ
  nesC は, 資源を有効利用するためイベント駆動を基に設計されている. しかし, イベントやコマンドによって呼び出された小さなタスクが並列に実行されるよう実装され, 処理の見通しが悪い.

3 視覚化機能を持つ開発環境

3.1 視覚化による利点

本研究では, nesC を用いた開発における問題点を改善するため, ソースコードとの同期が可能な視覚化機能を備えた開発環境を開発する. ソースコードとの同期は, ソースコードから図を生成する機能, 生成した図を編集することでソースコードへ変更を反映する機能から構成される. また, 生成した図に別の図やソースコードへの参照機能を持たせ, 図中の構成要素を選択することで参照先を容易に表示できるようにする.

これらの視覚化により, 開発期間の短縮を図る. コンポーネントの関連が複雑になる問題に対し, 開発環境が目的に合わせた図を提供することで, コンポーネント間の関連を整理し, 開発者の理解を支援する. 処理の流れの見通しが悪い点に関しても, イベント, コマンドの伝搬を図にまとめることで開発者に対し見通しのよい視点を提供することが可能である.

3.2 開発プロセスの改善

センサノード用のソフトウェアの目的は環境情報の取得であり, これ自体が大きく変化することはない. したがって, ソフトウェア開発においては既存のソフトウェアを再利用して開発する方が効率的である. ここでは, 既存のソフトウェアの機能を一部改変することでソフトウェアを開発する状況を想定する.

まず, 開発者は, 開発環境上でプロジェクトを作成し, 改変するソフトウェアの登録を含めた情報の設定を行う. 次に, 視覚化機能を提供するため, 開発環境はソースコードの解析を行う. 開発者は, 視覚化機能を利用すること

A Visualized nesC Development Environment for Understanding Software Structure
Takahiko Murakami†, Hideharu Matsuo††,
Kazuhisa Suzuki††, Yusuke Yokota†, and Eiji Okubo†
†College of Information Science and Engineering, Ritsumeikan Univ.
††Graduate School of Science and Engineering, Ritsumeikan Univ.

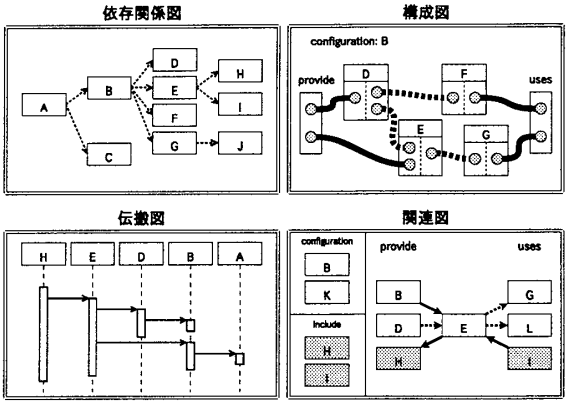

図 2　nesC コンポーネントの視覚化

により, ソフトウェア全体の構造を表示する図から構造を大まかに理解する. 必要な場合は, イベントやコマンドの伝搬を表現する図, およびモジュールのソースコードを参照しながら処理の流れを理解し, 改変を行う部分を特定する. 変更は, 構造を表現する図を編集し構造を変更したり, ソースコードを編集することで行う. 変更した部分は, イベント, コマンドの伝搬を図で表示することで視覚的に確認することが可能である.

3.3　視覚化の例

本研究で述べた視覚化機能の具体的な例を図 2 に示す. これらの図は, オブジェクト指向のモデリングで一般的に使われる UML を参考に作成を行った. 以下に, 主要な図について説明する.

依存関係図　ソフトウェアを形成する際に利用するコンポーネントの階層構造を表現する. 図の A から J はすべてコンポーネントである. 矢印は, コンフィグレーションからコンフィグレーションの内包コンポーネントへ向かって記されている. 図では, コンフィグレーション B が, コンポーネント D から G の内包コンポーネントで構成されていることを示している.

構成図　コンフィグレーションを図式化したものである. この図では, コンフィグレーション B について示している. 図で provide, uses と記述されている長方形は, B が提供または利用するインタフェースである. D から G は内包コンポーネントであり, それぞれのコンポーネントの左側に提供するインタフェース, 右側に利用するインタフェースが示されている. 太線で示されている接続は同等ワイアリングであり, 破線で示されている接続はリンクワイアリングである.

伝搬図　任意のイベント, コマンドから呼び出される処理の流れを表現する. この図では, コンポーネント H からのイベントの伝搬を示す. 図の H, E, D, B, A は, コンポーネントを示す. 矢印は, イベントの呼び出し元から呼び出し先へのイベントの伝搬を示し, 下位のコンポーネントから上位のコンポーネントへイベントが伝達されていく様子を示している.

関連図　任意のコンポーネントと関連のあるコンポーネントとその関係を図示する. この図では, コンポーネント E を中心とした関連を示す. 図中の configuration は, E を利用するコンフィグレーション, include は E の内包コンポーネントを示す. また, provide, uses は, E が提供する, または利用するインタフェースに接続されているコンポーネントを示す. 灰色で示されているコンポーネントは, E の内包コンポーネントを示す. E との接続は, 利用する側から利用される側へと矢印で表現されている. また, この矢印のうち, 破線のものはリンクワイアリングであり, そうでないものは同等ワイアリングを示す.

表 1　開発環境の機能比較

	※ 1	※ 2	※ 3	※ 4	※ 5
構文ハイライト	○	○	○	○	○
コードナビゲーション	×	○	○	○	○
入力補完	×	○	○	○	×
バージョン管理	○	○	○	×	×
ソースツリー管理	×	○	×	×	×
ビルド支援	○	○	○	○	×
エラー解析	○	○	○	○	×
エミュレーション環境	×	×	×	○	×
コンフィグレーションの視覚化	×	×	○	○	○
イベント, コマンドの視覚化	×	×	×	×	○
用途に合わせた視覚化	×	×	×	×	○
GUI エディタ	×	×	×	○	○

※ 1: TinyOS IDE　※ 2: TinyDT　※ 3: YETI
※ 4: TOSDev　※ 5: 提案開発環境

4　既存の開発環境との比較

既存の開発環境には TinyOS IDE [2], TinyDT [3], YETI [4], TOSDev [5] 等がある. 本研究が視覚化機能に重点を置いているのに対し, それぞれの開発環境は次のような目標を設定し, 開発が行われている (表 1 参照). TinyOS IDE は, 開発を単一の環境で行うことを目標としており, エディタからハードウェアへの書き込みまでの一連の機能が揃っている. TinyDT は, 構文解析による高度な入力補完機能を目標として開発されている. YETI は, nesC 開発の初心者と, 経験豊富な開発者それぞれの要求を調査し, どちらにも使いやすい環境を目指し, 開発環境の構築を支援する機能や, コンフィグレーションの図を利用したナビゲーション機能, Eclipse プラグインによるバージョン管理機能などの機能を持つ. TOSDev は, nesC の抽象化を開発環境で支援することにより, 開発効率を改善することを目的とする. このため, nesC のワイアリングを図を介して行うワイアリングエディタや, コンフィグレーションやソースコードを示すコードナビゲーションなどの機能を有する.

5　おわりに

本稿では, ソフトウェアの構造の理解を容易にするための視覚化機能を持つ nesC 言語の開発環境について述べた. 現在, Eclipse プラグインとして視覚化機能を実装している. 今後は, 提案手法による視覚化機能を用いたソフトウェア開発の機能評価を実施する予定である.

参考文献

[1] David, G., Philip, L., Robert, B,. Matt, W., Eric, B. and David, C.: The nesC language: A holistic approach to networked embedded systems, PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, pp.1–11 (2003).

[2] Richard, T.: Tinyos Eclipse Plugin by Richard Tynan, http://tinyoside.ucd.ie/

[3] Janos, S., Gyorgy, B. and Sebestyen Dora.: TinyDT, http://www.tinydt.net/

[4] Burri, N., Schuler, R. and Wattenhofer, R.: YETI: A TinyOS Plug-in for Eclipse, http://www.dcg.ethz.ch/publications/realwsn2006.pdf

[5] William, M. and Nigamanth, S.: TOSDev: a rapid development environment for tinyOS, Proceeding of the 4th international conference on Embedded networked sensor systems(SenSys '06), pp.387–388 (2006).