Samenvatting:

Rekenkrachtige apparaten hebben tegenwoordig toegang tot een rijke hoeveelheid aan informatie over hun omliggende omgeving, waargenomen door netwerksensoren en systeemmonitors. Dit laat toe om software-systemen te ontwerpen die zich aanpassen aan hun context, in tegenstelling tot meer traditionele systemen die veelal worden ontwikkeld in isolatie van omgevingsfactoren. Diensten aangeboden door dergelijke software-systemen kunnen zich aanpassen aan de waargenomen omstandigheden, waardoor deze diensten beter aangepast zijn aan de omliggende omgeving. Aanpassingen aan het gedrag van het systeem kunnen onaangekondigd plaatsvinden op om het even welk ogenblik. Indien niet met de nodige omzichtigheid wordt opgetreden, kunnen dergelijke aanpassingen daarom leiden tot inconsistenties in het gedrag van het systeem.
Om dergelijke inconsistenties te vermijden, moeten afhankelijkheden tussen mogelijke aanpassingen degelijk gedocumenteerd en zorgvuldig beheerd worden, om er aldus voor te zorgen dat de mogelijke interacties tussen deze aanpassingen niet onverenigbaar zijn voor de omliggende omgevingsfactoren.

Dit proefschrift onderzoekt welke garanties kunnen geboden worden omtrent de voorspelbaarheid van het gedrag van software-systemen die dynamisch kunnen aangepast worden tijdens hun uitvoering.
Op basis van een studie van verschillende dynamisch adaptieve software-systemen stellen wij de nodige vereisten op waaraan software-systemen moeten voldoen om de consistentie van gedragsaanpassingen van dergelijke systemen te garanderen.
We stellen een formele basis voor, genaamd contextuele Petri netten, ter ondersteuning van de ontwikkeling van consistente softwaresystemen in de aanwezigheid van dynamische gedragsaanpassingen. Deze formele basis voldoet aan de eerder opgestelde vereisten voor consistente dynamisch adaptieve software-systemen, en context georiënteerde programma's in het bijzonder, op drie niveaus: formalisering, uitvoering en analyse.

Contextuele Petri netten bieden een formalisering aan voor de definitie van dynamische gedragsaanpassingen, de interactie daartussen, en de notie van een consistent systeem in de aanwezigheid van dergelijke dynamische gedragsaanpassingen.
Interacties tussen aanpassingen worden geformaliseerd door welbepaalde regels die enerzijds de intentie van een programmeur op een hoog niveau weten te vatten, en anderzijds een representatie op laag niveau bieden die een automatische verificatie van de regels toelaten.
Verificatie van de consistentie van een dynamisch adaptief software-systeem kan dan worden voorzien op twee niveaus.
Enerzijds kunnen tijdens het ontwerp en de ontwikkeling van het systeem bepaalde systeemeigenschappen geanalyseerd worden, om in een vroeg stadium mogelijke incoherenties in de definitie van interacties tussen aanpassingen te identificeren.
Anderzijds kunnen tijdens de uitvoering van een dergelijk systeem de gedefinieerde interactieregels tussen aanpassingen continue geverifieerd worden om aldus te garanderen dat er zich geen inconsistenties zullen voordoen.
Met deze voorgestelde formele basis als fundament bieden we een hulpmiddel aan voor het ontwerp, het beheer de de simulatie van dynamische aanpassingen en hun interacties.

Dit werk wordt gevalideerd door het nut aan te tonen van de aanpak bij het analyseren van bestaande context georiënteerde toepassingen, door de grenzen van context georiënteerd programmeren te verruimen, en door de uitbreidbaarheid door de formele basis te illustreren.

English

Computing devices now enable access to rich information about their surrounding execution environment gathered through sensor networks or system monitors. This ability allows software systems to be conceived with context in mind, instead of being created in isolation as in traditional approaches for software development. Services provided by software systems can be adapted to sensed conditions rendering such services more appropriate to the surrounding execution environment. Adaptations to the system's behavior take place unannounced over time. However, if not dealt with carefully, the behavior provided by such adaptations could lead to inconsistencies in the system's behavior. In order to avoid such inconsistencies, dependencies between adaptations must be carefully managed so that interactions gathered for the surrounding execution environment are not rendered incompatible.

This dissertation investigates how to provide more guarantees about the predictability of the system's behavior when it is adapted dynamically at run time. Based on the observation of different dynamically adaptive software systems, we put forward a set of requirements that software systems should satisfy to ensure consistency of its behavioral adaptations. We propose a formal basis to support the development of consistent software systems in the presence of dynamic behavioral adaptations, called context Petri nets. This formal basis complies with the requirements for consistent dynamically adaptive software systems, and in particular context-oriented programming, on three levels: formalization, execution, and analysis.

Context Petri nets offer a formalization for the definition of adaptations, the interactions between them, and the notion of consistency of a system in the presence of dynamic behavioral adaptations. Interactions between adaptations are formalized by a well-defined set of rules that capture the intention of programmers at a high-level, while enabling the low-level representation and automatic verification of those rules. Consistency verification of the system is provided at two levels. At design-time, system properties can be analyzed for the identification of possible incoherence in the definition of interactions between adaptations. At run-time the satisfiability of all interaction rules between adaptations is verified, hence, it can be ensured that no inconsistencies occur. Based on the proposed formal basis, we offer a tool for the design, manipulation, and simulation of adaptations and their interactions. This work is validated by demonstrating its usefulness in analyzing existing context-aware applications, its appropriateness in broadening the frontiers of context-oriented programming, and its extensibility by expanding the formal basis itself.

Cardozo Nicolas