Appendix A

In gewoon Nederlands

Het is natuurlijk niet mijn bedoeling om een ondoorgrondelijk proefschrift te schijven. Ik vind het belangrijk om ook aan niet-wetenschappers duidelijk te maken waar ik me tijdens mijn promotie mee bezig heb gehouden. Dat doe ik hier. Ik heb gekozen voor alledaagse voorbeelden, omdat dat veel sprekender is dan wanneer ik op de gebruikelijke wijze termen uit de informatica uitleg: we gaan het spel en de regels van monopoly proberen te beschrijven in termen van mijn werk.

A.1 Een berg informatie

Een spel monopoly kun je zien als een grote berg informatie. Bijvoorbeeld de hoeveelheid geld in het bezit van elke speler, wie bezit welke straten, wie is er aan zet, wat is de volgorde van de vakjes op het bord, ... Informatici brengen in die berg informatie graag wat structuur door het in kleinere hoopjes bijeen te vegen, en te kijken hoe die hoopjes met elkaar gerelateerd zijn.

Om de draad niet kwijt te raken is het handig hoopjes informatie te maken voor elk 'ding' in het spel: een hoopje voor de informatie van elke speler (naam, banksaldo, ...), een hoopje voor elke straat, eentje voor elk huis, elk hotel, ... En dan zijn er relaties tussen de nu overzichtelijke hoopjes informatie: een speler heeft een pion op een bepaalde straat, een straat kan met hotels bebouwd zijn, ...

Zo, dat ruimt op. Alle informatie is nu gerangschikt en in relatie tot elkaar gebracht. Elk hoopje informatie heet in computertermen een *object*, en een relatie heet een *link*. Als je nu voor elk object een kaartje maakt dan kan het zo in een kaartenbaksysteem worden gestopt... of in de computer natuurlijk.

Helaas zijn de spelregels veel minder overzichtelijk. Als speler Rick met de dobbelstenen 3 gooit, dan zie je op het bord zó wel wat er gebeurt (alhoewel, houd wel je straten in de gaten!) maar in een kaartenbaksysteem moet je toch flink zoeken voor je alle kaarten hebt bijgewerkt. In een computer gaat het netzo als in de kaartenbak, dus ook dat is tamelijk bewerkelijk.

Om dat spitten in de berg kaartjes duidelijker te maken worden ook de beschrijvingen van 'dingen om te doen' in kleinere brokjes beschreven. Die brokjes heten in informaticatermen *processen*. Het is gebruikelijk om een apart proces te beschrijven voor elk object zoals we dat al hadden gevonden.

A.2 Stapje voor stapje

Als we willen weten hoe monopoly er in termen van processen uitziet, dan kunnen we het beste kijken welke losse stapjes je tijdens het spel neemt, en later kunnen we dan kijken welke volgordes van zulke stapjes toegestaan zijn.

Een voorbeeld. Stel, speler Rick gooit 3. Zulke gebeurtenissen schrijf ik in mijn proefschrift op als worp(Rick,3); de *naam van de gebeurtenis*, hier worp, geeft aan wat voor soort gebeurtenis dit is, de rest is aanvullende informatie.

Nou blijft het niet bij die ene gebeurtenis, want er wordt een reeks andere gebeurtenissen ontketend, die denkbeeldig allemaal *tijdens* de worp gebeuren. Misschien is de worp eigenlijk wel een dubbelworp, en dan gebeurt die event gewoon óók, tegelijkertijd. En, de dingen die uit worp(Rick,3) kunnen volgen zijn bijvoorbeeld verlaat(Rick,CoolSingel) (Rick verlaat de Coolsingel) en aankomst(Rick,LeidscheStraat). En als de Leidschestraat eigendom van iemand anders is, dan volgt er ook nog een betalings-gebeurtenis (snif). De stapel gebeurtenissen die elkaar zo veroorzaken vormen samen een elementaire actie in ons monopolyspel.

Het afhankelijk zijn van andere dingen elders in het spel, is één van de punten waarop mijn aanpak zich onderscheidt van de gangbare processen voor objecten: ik knoop alle betrokken processen (de lopende speler, het aankomstvakje en de eigenaar daarvan) aan elkaar en laat ze samenwerken. Dat kan ik doen doordat ze eigenlijk allemaal dezelfde gebeurtenissen te zien krijgen, en ze nemen elk op hun eigen houtje actie als gevolg daarvan.

Er zijn best veel gebeurtenissen van belang in zo'n monopolyspel, maar die mogen natuurlijk niet zomaar in elke volgorde gebeuren. Daarom leggen mijn processen, wederom stringenter dan soortgelijke practische modellen, beperkingen op aan welke gebeurtenissen mogelijk zijn, en in welk volgorde. Bijvoorbeeld, de volgorde van wie er gooit moet netjes vastgelegd worden, eerst Rick, dan Maarten, en dan weer overnieuw. Een ander voorbeeld is als Maarten in de gevangenis zit, die mag daar pas uit als hij dubbel gooit. Dus als Maarten in de gevangenis zit dan is hij in een speciale toestand, dus worp(Maarten,5) zonder gelijktijdige dubbelworp mag nooit samengaan met verlaat(Maarten,Gevangenis) (aannemende dat er niet betaald is, of een kaart 'verlaat de gevangenis zonder betalen' is ingezet).

Ik heb een notatie bedacht waarin je al dat soort dingen kunt opschrijven. Een proces wordt getekend in een diagram met een rechthoek eromheen en de naam bovenin; afgeronde rechthoeken geven de 'toestanden' van het proces weer, en de pijlen ertussen geven aan hoe een gebeurtenis een proces van de ene toestand in de andere brengt. Het bijzondere van mijn notatie zit in de notaties bij de pijlen, en hun precieze betekenis.

A.3 De eerste stapjes zijn ook gewoon stapjes

De processen in mijn systemen reageren op gebeurtenissen, en verder is er niets. Maar soms moet een nieuw proces (plus object, of hoopje informatie) worden aangemaakt voor bijvoorbeeld een nieuw huis dat je aanschaft, of als een speler failliet gaat dan moet zijn object worden weggegooid.

Dat is in mijn processen simpel maar ook krachtig opgelost: het aanschaffen van een nieuw huis is voor de meeste processen een gewone gebeurtenis, maar voor huizen is het speciaal, want het maakt een nieuw huis aan. Alle gebeurtenissen zijn dus gewone gebeurtenissen, maar processen behandelen bepaalde gebeurtenissen speciaal omdat die als eerste of laatste in een proces genoemd staan.

Ook in deze aanpak onderscheid ik me van bestaande practische notaties. Dat is echter bittere noodzaak. Als ik processen aan elkaar wil knopen en er ook nog dingen over wil bewijzen, dan kan ik maar beter zorgen dat ook het aanmaken van een nieuw proces gewoon meedoet in de processen.

Overigens interessant op te merken is dat mijn processen aan reïncarnatie doen: als een huis wordt verwijderd (bijvoorbeeld bij inruilen voor een hotel) dan gaat'ie terug in de doos, klaar voor de volgende aankoop, en netzo gebeurt het ook met mijn processen.

A.4 Waarom het zo precies moet

Als ik een weddenschap of een spelletje monopoly met Maarten aanga, dan gaat het niet om onbelangrijke dingen als geld, maar dan gaat het om dingen van Groot Belang, zoals marsrepen. Na een poosje op die manier Maarten's secundaire arbeidsvoorwaarden verzorgd te hebben werd ook mij duidelijk dat het belangrijk was de spelregels heel goed vast te leggen, opdat het allemaal eerlijk zou verlopen. Ook kinderen hebben daar baat bij, ze gaan anders vechten.

Hoe vreemd het ook mag klinken, soortgelijke problemen en ruzietjes komen bij het maken van een groot ontwerp in software evenzeer voor, alleen dan met een stropdas voor. Het probleem blijft

dat onduidelijkheid leidt tot misverstanden, en soms kunnen die duur uitpakken. Dus ook in de informatica is het wel zo handig om precies te weten waar je het over hebt. Het is dan ook verrassend te merken dat de meeste informatici schuw zijn van precisie als ze diagrammen tekenen.

Voorbeelden van dingen die door gebrekkige precisie fout kunnen gaan schuilen vaak in de uitzonderingen en de randgevallen. Mag speler Rick bijvoorbeeld de Kalverstraat kopen als speler Maarten die al gekocht heeft, en zo ja, moet Rick dan nog huur betalen aan Maarten? Dat soort dingen worden duidelijk beantwoord in de wiskundige versie (omdat die van nature compleet is) maar staat het ook in de papieren regels? Die gaan er van uit dat je weet wat 'in bezit van' betekent. Allemaal mooi en wel, maar een computer heeft dit soort algemene ontwikkeling niet, en het moet dus allemaal expliciet worden opgeschreven.

Natuurlijk zijn er al heel lang wiskundigen die werken aan precisie, maar die verliezen soms de practische waarde van een theorie uit het oog, omdat de problemen daarzonder ook al moeilijk genoeg zijn. Er waren dus twee werelden die bezig waren met processen, maar ze kwamen maar niet bij elkaar. Ik heb dat met deze promotie voor elkaar gekregen, door op de kennis en ervaring in die twee werelden voort te bouwen. Ik heb een practisch bruikbare notatie bedacht, en daar heb ik een wiskundig preciese definitie van gegeven, zodat de informaticus diagrammen kan tekenen met een precies vastgelegde betekenis.

A.5 Soms weet je het gewoon niet

We kunnen nu al veel van het monopolyspel in processen en objecten vastleggen, maar we moeten ook weer niet te ver gaan. Bijvoorbeeld, de acties ten gevolge van het trekken van een kanskaart zijn totaal willekeurig, en daardoor kun je gewoon niet zeker zijn wat er gebeurt als gevolg van het trekken van een kanskaart.

Wiskundigen kennen dit probleem al heel lang, dat een keuze valt, maar het is onbekend welke. De term daarvoor is *non-determinisme*, en het is in de informatica nog niet gebruikelijk om zulk bekend gebrek aan kennis op te schrijven.

Omdat ik de computer bewijzen wil laten leveren, moet ik dat hier wel doen. Want een bewijs is niet veel waard als het alleen geldt wanneer 'ga verder naar Kalverstraat' boven op de stapel ligt. Je wilt juist dat een bewijs altijd geldt, ongeacht wat de volgende kanskaart zal zijn!

A.6 Bewijzen hoe het zit

Het ontwerpen van een complex stuk software blijft mensenwerk, en er kunnen fouten in komen. Het is daarom nuttig om checks in te bouwen om te zien of alles wel klopt, vergelijkbaar met de balans die moet kloppen in een boekhouding. Zijn de uitkomsten daarin ongelijk, dan heb je ergens een fout gemaakt.

Ook voor ontwerpen kun je controles inbouwen, maar dan niet met getallen. Processen zijn gelijktijdige gebeurtenissen, en daarvan meerdere in bepaalde volgordes. En elk daarvan kan iets veranderen aan de status en gegevens van een programma.

Het handige aan de wiskunde is niet alleen dat het zo precies is, maar bovendien kun je een wiskundige uitspraak op allerlei manieren anders opschrijven zonder dat het opeens niet meer klopt, en als je geluk hebt kun je daar zelfs zo ver mee gaan dat je een gewenste eigenschap kunt bewijzen. Als dat fout loopt, dan heb je een fout in je ontwerp of in je eis zitten, net als bij de balans die niet klopt. Ik doe dit in mijn onderzoek met een bewijsprogramma, zeg maar een wiskundige-in-een-doosje, zodat niemand het moeizame bewijswerk met de hand hoeft te doen maar dat het aan de computer kan worden overgelaten, want anders zou (bijna) niemand er mee willen werken.

A.7 Verfijnde kneepjes

Een extra winstpunt van een wiskundige ondergrond is dat er een heleboel zinnig onderzoek ter beschikking komt voor practische toepassingen. Zo is er bijvoorbeeld de mogelijkheid om globaal beschreven stappen in een proces in meer detail uit te werken; in de wiskunde heet dat *refinement*.

In het monopolyspel zien we dat ook terug. Bij de actie worp(Rick,3) gaat Rick's pion 3 plaatsen vooruit, maar eigenlijk gebeurt dat in drie losse stapjes achter elkaar, en samen noem je dat dan 3 plaatsen vooruit gaan. De uitwerking is dan iets als stapje(Rick,Spui,Plein), dan stapje(Rick,Plein,Waterleiding) en tenslotte stapje(Rick,Waterleiding,LangePoten). Aan de eerste plak je dan bovendien de gebeurtenis verlaat(Rick,Spui) vast, en aan de laatste aankomst(Rick,LangePoten).

Dit is voor een simpel geval als dit nog wel te overzien, maar ook hier geldt dat mensen fouten kunnen maken. Net als je allerlei algemene eigenschappen kunt testen met wiskunde, kun je ook kijken of een bepaalde detailuitwerking wel klopt met de regeltjes voor detailering, tenminste als je die regeltjes kent. En dat treft: die regeltjes, daar hebben wiskundigen al veel werk in gestoken en daar kun je dus de vruchten van plukken.

Refinement is voor complexe ontwerpen van groot belang. De meeste ontwerpmethoden passen iets dergelijks toe, maar dat lijkt al erg veel op programmeren, zodat er opeens allerlei extra details nodig zijn, en sommige van die extra details zijn ongewenst omdat ze de aandacht van het ontwerpen afleiden. Dat is niet de bedoeling, omdat je dan het overzicht weer kwijt raakt, en daar was het nou juist allemaal om begonnen!

A.8 Alles of niets

Het zal inmiddels duidelijk zijn, de simpele gebeurtenis worp(Rick,3) leidt tot een hoop andere gebeurtenissen die op allerlei manieren verweven kunnen zijn. Soms kom je er halverwege een worp achter dat je de geplande acties toch niet had kunnen doen. Denk maar aan een speler die een fout heeft gemaakt, of die blijkt niet te kunnen betalen voor een hotel dat al op zijn straat gezet is. De beste oplossing is dan om de gehele worp terug te draaien. Niet alleen een paar, maar *alle* gelijktijdige gebeurtenissen.

Dit kan in een computerprogramma met iets dat als *transactie* bekend staat. Transacties zijn afkomstig uit de wereld van de *databases*, en dat zijn eigenlijk electronische kaartenbakkensystemen, dus daar kan alle informatie in worden opgeslagen; zulke systemen laat ik ook automatisch construeren door de computer; dat kan dankzij de wiskundige precisie van de processen. Transacties zeggen (onder andere) dat een stapel veranderingen van de informatie helemaal wel, of helemaal niet gebeurt; het is alles of niets.

Het is kenmerkend aan mijn werk hoe gemakkelijk transacties er in passen. Alternatieve aanpakken werken niet met gebeurtenissen die gelijkertijd optreden, en dat bemoeilijkt dit enorm. Door mijn eenvoudiger (maar even krachtige) aanpak voorkom ik een hoop problemen waarvoor niet echt een nette oplossing bestaat.

A.9 Volledigheidshalve...

Het is al heel veel werk om bewijzen te kunnen leveren voor processen die links naar elkaar hebben. Ik heb me in mijn onderzoek daartoe beperkt. Bijvoorbeeld het rekenen aan de prijzen die je moet betalen als je op een straat met een hotel aankomt, dat beschrijven mijn processen niet. De bedragen kun je gewoon doorgeven bij een gebeurtenis, maar mijn processen rekenen niet aan die waarden, ze geven ze alleen maar door.

In onderzoek maak je vaak zulke versimpelingen, iemand anders kan er later immers op voortbouwen. In wezen is dat wat ik ook gedaan heb, ik heb heel veel bestaande wiskunde en informatica kunnen gebruiken en voeg daar een stukje van mezelf aan toe.