Management arrangements, Teaching, Business Plan
Promotion + Marketing

Summer School – update

18 JAN 2011

Tasks to be done

1. **Draft and seek approval for revised business plan.**

- Develop a budget in consultation with finance in a format to go into the five-year plan
- Decide where the budget will be held; the working assumption is in CLL
- If necessary submit budget plans to Finance and General Purposes Committee ahead of the five-year planning process: we need confirmation that the budget is agreed by 31st March if we are to go ahead in 2012.
- Update/draft narrative business plan to International Committee on 26th January.
- Full business plan to International Committee for sign-off on 2nd March.
- Completion of course approval form for the Summer School
- http://www2.warwick.ac.uk/services/academicoffice/quality/categories/courseapproval/course/ **(IO to complete)**
- (BP to Senior Officers on 4th March)?
- BP to Steering before the end of March for sign-off.

2. **Design modules and have them approved.**

- Departments to be given clear guidance on format required for module descriptions (for academic approval and for marketing) at meeting of Summer School group in mid-January.
- http://www2.warwick.ac.uk/services/academicoffice/quality/categories/courseapproval/module/ **(each department to complete)**
- Modules submitted for approval by mid-February
- Modules approved by mid-March
- Descriptions of modules for marketing ready by 1st March

*

3. **Establish management arrangements**

- Establish Summer School Working Group in mid-January, to consist of representatives of departments offering modules, IATL, CLL and International Office. Interim Chair: Director of the International Office. Pro-Vice-Chancellor, Education and Student Experience to be consulted about permanent Chair. Meetings to be held as required.
- Departmental responsibilities to be clearly defined; suggestions:

*

Academic departments	Module design, teaching and assessment
IATL	Academic guidance to ensure excellence and

~8k hrs

1st / 2nd / 3rd

Visa probs if newly graduated for UK international students.

	innovation in teaching
CLL	Admissions and enrolment; budget holder
International Office	Approval documentation Marketing and promotion

- Agree non-teaching staffing requirements.

4. Develop a marketing plan

– bursaries

- Consult with Communications Department
- Ensure consistency and attractiveness of the offer
- Establish prominent web-presence
- Develop electronic and print materials
- Write a communications plan, to cover existing contacts (partner universities, alumni, current students), conferences and fairs, media opportunities
- Agree soft-launch activities prior to hard launch at NAFSA at end of May

(US) Nat. Assoc. Foreign Stud Advisors

5. Confirm accommodation, food and other social arrangements

- Student accommodation (60 rooms) booked
- Teaching accommodation confirmed by academic departments as part of their module offer
- Food arrangements for students to be agreed
- Social activities plan (including trips and on-campus activities) prepared
- Agree role of student hosts/helpers

Constructivist computing | for Interdisciplinary ~~Apps~~ Modelling

|| IGGY goes virtual

4 modules 15 each

James Kennedy

WBS

Steve Hindle, History

English History (Modern)
Shakespeare

Nick Monk IATL

(Virtual) Approval by end of May

What Module Proposal Form is needed?

— devolved to department? Module cats in Dept. deadline??

tweak for promotion purposes ... stipulate lang. reqts.

Constructivist Computing for Interdisciplinary Modelling

Principal Module Aims

Constructivist computing is a conceptual framework for computing that is broader in scope than traditional 'computational thinking'. The overall aim of this module is to convey the potential for constructivist computing to account for and to enhance computing practice even in areas beyond the remit of computational science. Constructivist computing is based on the principles, tools and potential applications of Empirical Modelling (EM), a new approach to modelling and computing that has been developed by staff and students here in Computer Science at Warwick over many years. Students will be: (a) instructed in fundamental EM concepts and techniques; (b) shown how EM is informed by perspectives relating to computer science, philosophy and constructivist thinking; (c) introduced to the fundamental notion of construal as it applies to key application areas; and (d) participate in collaborative modelling studies directed at topics particularly relevant to their personal interests and disciplinary specialisms.

Module Overview

The module will be delivered over three weeks, each week (comprising 5 days) representing a different phase of teaching and learning activity. In broad terms, Week 1 will motivate constructivist computing and introduce the basic principles, concepts and tools, Week 2 will be an in-depth study of papers and models relating to specific application areas for constructivist computing, and Week 3 will be devoted to group-based collaborative modelling studies focusing on a core theme that will be advertised in advance of the module.

An example of a suitable core theme might be modelling the musical and dramatic production and appreciation of Mozart's opera *The Magic Flute*. The model-building activity relevant to such a theme would be broad, and draw on several disciplines. For instance, model-building could address the structural, thematic and harmonic aspects of the musical score, the characteristics of the cast (e.g. vocal range and physical characteristics) both as given and as ideally required, the layout of scenes and cast members involved in them, the design of costume and scenery, the movement of singers as planned by the artistic director, characteristics of specific performances such as relate to choice of tempi and potential cuts, the libretto and narrative, and the logistics (e.g.) of timetabling rehearsals and organising scene changes during a performance. Such a modelling exercise would draw on skills from computing, music, drama, business, design and education. (Note that the quality of the modelling that can be done with respect to any particular aspect may not compare with special-purpose professional tools, but the character of the modelling is distinctive and untypical of traditional computer models. Particularly significant is the flexibility, openness to reinterpretation and extension, and the way in which different aspects of the modelling are integrated and can be reconfigured.)

Other possible core themes might engage with different disciplines. They might include: educational applications for teaching mathematics; accessible models of medical knowledge; modelling in support of mathematical research; modelling for business; modelling relating to literature and archaeology in the humanities; modelling for geographical applications. Rather than having a single core theme for each summer school, it may be appropriate to have two complementary themes e.g. one representative of science and the other of arts and the humanities.

Outline syllabus

The first week will introduce the notion of *constructivist computing*, as supported by Empirical Modelling (EM). The key notion of *a construal* will be introduced and discussed in relation to modelling and programming. The key concepts of observables, dependencies and agents will be explained and illustrated with standard examples. The nature of applications characteristic of constructivist computing will be explored, with reference to topics such as learning technologies, personalised applications, models for medical use, model-building to support experimental science or mathemetical research, business modelling and humanities computing. An appropriate philosophical orientation for constructive computing will be discussed with reference to critiques of the "rationalistic" account of computing, such as that discussed in Winograd and Flores, Latour's proposals for rehabilitation of the concept of construction in his paper *The Promises of Constructivism*, William James's radical empiricism and other topical philosophical stances.

In keeping with the spirit of constructivist computing, practical study of models will be an essential strand of the module. The tools to be introduced will include the Cadence and EDEN interpreters and auxiliary environments such as the EM presentation environment (EMPE) and the dependency modelling tool (DMT).

Week 1 will involve ten lectures and ten laboratory sessions. Through a personal consultation with each student, we shall identify their particular strengths and interests and set up project groups to address different aspects of the core theme in Week 3.

Week 2 will look in more detail at specific applications of EM relating to the chosen core theme, as represented by papers and model drawn from the EM archive. Each of the five days will be devoted to in-depth study of one or more existing models devoted to a specific theme (cf. the range of different kinds of observables, dependencies and agents represented in modelling strands associated with the illustrative core theme mentioned above, for which there are precedents in the EM archive in the form of construals of musical compositions, timetabling models, visualisations and animations etc). Relevant key papers drawn from the EM archive will be studied in parallel. During this week, students will work on exercises in analysing and documenting models, typically making use of special-purpose tools such as the EMPE and the DMT.

In Week 2, there will be one or two introductory lectures and one or more sessions involving the analysis and demonstration of models relating to each aspect of the core theme. For the rest of the week, students will work in laboratory sessions alongside a small team of Warwick assistants (preferably undergraduates familiar with EM principles and tools who ideally would have themselves already worked on the modelling exercises and prototyped models related to the core theme).

Week 3 will be devoted to group work aimed at documenting and enhancing existing prototype models and conceiving and/or developing new models addressing the core theme. In the constructivist spirit, practical modelling activity will play a central role in this; even the *conception* of models will involve some experiments in model-building. An additional component of this activity that involves no conceptual shift within the EM framework might be enhancement of the modelling tools better to address the specific demands of the core theme.

The assessment for the module will be based on laboratory work in Week 1, the analysis and documentation exercise in Week 2, and the contribution to the group work in Week 3, as expressed through practical work and assessed through group presentations and individual oral examinations to be held at the end of the module.

To complement the above teaching programme, we shall organise one or two seminars from external speakers with specialist expertise relating to the core theme, such as an expert in humanities computing or former EM doctoral students who have worked on (e.g.) collaborative group work using EM. We shall also engage other teaching assistants (e.g. from the Warwick Student Opera Group) to enact authentic activities relating to the core theme that help to illuminate the processes involved in the application (such as a "blocking rehearsal" for an ensemble from the Magic Flute, at which the precise movements and interactions of the singers are planned out in detail).

Supplementary issues

The notion of constructivist computing stems from the Empirical Modelling (EM) research project. The scope of EM is reflected in the outline syllabus for the 4th year module CS405 Introduction to Empirical Modelling - taught by Meurig Beynon and Steve Russ over the last 9 years - as set out below. CS405 is a 15 CATS module comprising 20 lectures and 10 two hour laboratory sessions that is examined by coursework and a 3 hour examination each weighted at 7.5 CATS. There are rich online resources for CS405 (see the module webpage at http://www2.warwick.ac.uk/fac/sci/dcs/research/em/teaching/cs405/ for which the password if 'radicalempiricism') and these are complemented by the EM website at http://www.dcs.warwick.ac.uk/modelling/ which features over 100 publications and 175 models in the EM archive.

Whilst the ideas behind constructivist computing are well-suited for appreciation by students who have no knowledge of traditional computer science and lack technical computing skills, it is unclear to what extent such students will be able to engage with practical modelling activities without some support from assistants with EM expertise. For this reason, the viability of the module may depend on being able to recruit students with suitable expertise in EM. There are some logistic issues to be addressed here since EM is only taught in the 4th year of the undergraduate programme and few 4th year students can guarantee availability in the vacation after the end of their degree studies. The students probably best suited to the teaching assistant role are those who have done EM projects in their third year and it is unclear how such students can be supervised when neither Beynon nor Russ is a salaried Computer Science staff member.

Appendix

Outline syllabus for *CS405 Introduction to Empirical Modelling*

The precise structure and syllabus of the module will vary from year to year. The key themes which will be represented are:

- **Motivating ideas and orientation:** The study of Empirical Modelling is motivated by dissatisfaction with the account of computing-in-the-wild that can be given by classical computer science. Important issues in computing cannot be addressed by invoking logic and algorithmic thinking alone. Relevant critiques include: Brian Cantwell-Smith on the foundations of computing; Kent on the limitations of abstract data representations; Winograd and Flores on groupware design; Brookes on the challenges of software development; Jackson on the limitations of formal specification; Naur on the essential role of intuition in system development; McCarty on the nature of 'humanities computing'. Relevant practical developments include: the application and extension of spreadsheet principles, both in educational software (e.g. dynamic geometry packages) and in software development tools (e.g. Flex and the Windows Presentation Foundation framework), and the continuing interest in how to resolve the problems of ensuring that software matches its requirements (e.g. as represented in the motivating ideas behind object-oriented programming, the work of Harel on participatory design, and agile development methods).

- **Conceptual framework:** Classical computer science gives an excellent account of scientific applications of computing technology in which the role of the computer is to compute results on the basis of an established mathematical theory. In many practical applications of computers, the emphasis is different: the computer is used to construct artefacts that aid understanding (resembling what the philosopher of science David Gooding terms construals). Sense-making proceeds in parallel with the making of such construals in a constructivist idiom. Activities of this kind cannot be satisfactorily described within a 'rationalistic' logicist framework. Empirical Modelling focuses on the way in which understanding is developed through observation and experiment, prior to the emergence of established concepts that can be expressed in a formal language. Its philosophical stance is pragmatic in the spirit of William James's radical empiricism.

- **Concepts and principles:** The key concepts of Empirical Modelling are observables, dependencies and agents. Where classical computer science seeks a foundation in formal semantics, Empirical Modelling appeals to the Jamesian idea that all knowing is rooted in the connections that are themselves given in experience, in the way for instance that images and ideas come to mind in the very act of reading symbols on a page. Empirical Modelling artefacts can evolve seamlessly from construals to models to programs in ways that reflect the transition from the personal and subjective to the public and objective. Such artefacts can be concurrently manipulated from the perspectives of many human agents so as to achieve conceptual integrity; objects, processes and systems - potentially with formal semantics - emerge. The LSD notation has been developed as a way of giving an account of this activity of rationalisation and objectification.

- **Tools and techniques:** To date, Empirical Modelling has been principally supported by the EDEN interpreter and its variants (command line, distributed, web-enabled).

EDEN was introduced as an evaluator for "definitive (definition-based) notations". Definitive notations provide a means to express dependencies between observables of different kinds, generalising the types of relationships that can be established between values in the cells of a spreadsheet. Environments constructed using EDEN are unlike conventional programs: they typically have the same provisionality, openness and messiness that is characteristic of our partial personal understanding of experience. Tools for Empirical Modelling have recently been significantly enhanced by the development of an interpreter that attacks the same goal of supporting flexible modelling of personal understanding by exploiting dependency and object-orientation. Cadence is a research prototype which complements (and may eventually subsume) model-building with EDEN, giving support for the emergence of objects and processes that is characteristic of mature models. Other supporting tools include the Dependency Modelling Tool and the Abstract Definitive Machine.

- **Applications:** Empirical Modelling has potential applications in many areas, including educational technology, computer-aided design, software development, humanities computing, visualisation, games development and concurrent systems simulation. These are illustrated in the many models that can be found in the Empirical Modelling projects archive. They have also been the subject of graduate student research, as documented in the theses that can be accessed from the Publications link on the Empirical Modelling webpage.

There is scope to study the module in several ways, with many different kinds and levels of background experience. For the advanced student, there are good prospects for contributing to Empirical Modelling research through dissertation work. It is also possible to follow the module with little prior knowledge of computing.

CS405 Introduction to Empirical Modelling

THE UNIVERSITY OF WARWICK

Academic Aims

The module introduces students to the principles, tools and potential applications of *Empirical Modelling (EM)*, a new approach to computing that has been developed by staff and students here in Computer Science at Warwick over many years (see the EM website). The module offers a complementary perspective on computer science broader in scope than 'computational thinking', and is oriented towards aspects of computing practice for which formal methods offer limited support. These include activities such as conceptual design, requirements cultivation and exploratory modelling that involve close human-computer co-operation and often engage many human participants concurrently. The most appropriate areas of application for EM are those in which sense-making and model-building develop in parallel, as in a constructivist style. They include computer support for learning, decision support and creative design in business, engineering and the humanities.

Learning Outcomes

The module promotes awareness of the fundamental and distinctive role that empirical knowledge plays in conceiving and implementing computer-based systems. It also teaches practical skills that are relevant to the individual and team work that typically precedes the explicit specification and design of such systems. Over and above this, it introduces ways of thinking about computing and ways of using computers that are topical in relation to current and emerging technologies and applications.

Content

The module involves 40 contact hours, comprising 20 one-hour lecture sessions and 10 two-hour sessions of practical laboratory work. Assessment will be 50% by the submission of a short paper and an associated documented modelling exercise to the annual edition of the *Warwick Electronic Bulletin* early in Term 2, and 50% by a three hour examination in Term 3.

The precise structure of the module varies from year to year. The key themes which will be represented are:

- **Motivating ideas and orientation:** The study of Empirical Modelling is motivated by dissatisfaction with the account of computing-in-the-wild that can be given by classical computer science. Important issues in computing cannot be addressed by invoking logic and algorithmic thinking alone. Relevant critiques include: Brian Cantwell-Smith on the foundations of computing; Kent on the limitations of abstract data representations; Winograd and Flores on groupware design; Brookes on the challenges of software development; Jackson on the limitations of formal specification; Naur on the essential role of intuition in system development; McCarty on the nature of 'humanities computing'. Relevant practical developments include: the application and extension of spreadsheet principles, both in educational software (e.g. dynamic geometry packages) and in software development tools (e.g. Flex and the Windows Presentation Foundation framework), and the continuing interest in how to resolve the problems of ensuring that software matches its requirements (e.g. as represented in the motivating ideas behind object-oriented programming, the work of Harel on participatory design, and agile development methods).
- **Conceptual framework:** Classical computer science gives an excellent account of scientific applications of computing technology in which the role of the computer is to compute results on the basis of an established mathematical theory. In many practical applications of computers, the emphasis is different: the computer is used to construct artefacts that aid understanding (resembling what the philosopher of science David Gooding terms *construals*). Sense-making proceeds in parallel with the making of such construals in a *constructivist* idiom. Activities of this kind cannot be satisfactorily described within a 'rationalistic' logicist framework. Empirical Modelling focuses on the way in which understanding is developed through observation and experiment, prior to the emergence of established concepts that can be expressed in a formal language. Its philosophical stance is pragmatic in the spirit of William James's *radical empiricism*.
- **Concepts and principles:** The key concepts of Empirical Modelling are *observables*, *dependencies* and *agents*. Where classical computer science seeks a foundation in formal semantics, Empirical Modelling appeals to the Jamesian idea that all knowing is rooted in the connections that are themselves given in experience, in the way for instance that images and ideas come to mind in the very act of reading symbols on a page. Empirical Modelling artefacts can evolve seamlessly from construals to models to programs in ways that reflect the transition from the personal and subjective to the public and objective. Such artefacts can be concurrently manipulated from the perspectives of many human agents so as to achieve conceptual integrity; objects, processes and systems - potentially with formal semantics - emerge. The LSD notation has been developed as a way of giving an account of this activity of rationalisation and objectification.
- **Tools and techniques:** To date, Empirical Modelling has been principally supported by the EDEN interpreter and its variants (command line, distributed, web-enabled). EDEN was introduced as an evaluator for "definitive (definition-based) notations". Definitive notations provide a means to express dependencies between observables of different kinds, generalising the types of relationships that can be established between values in the cells of a spreadsheet. Environments constructed using EDEN are unlike conventional programs: they typically have the same provisionality, openness and messiness that is characteristic of our partial personal understanding of experience. Tools for Empirical Modelling have recently been significantly enhanced by the development by Nick Pope of an interpreter that attacks the same goal of supporting flexible modelling of personal understanding by exploiting dependency and object-orientation. Cadence is a research prototype in which Pope's interpreter complements (and may eventually subsume) model-building with EDEN, giving support for the emergence of objects and processes that is characteristic of mature models. Other supporting tools include the Dependency Modelling Tool and the Abstract Definitive Machine.

- **Applications:** Empirical Modelling has potential applications in many areas, including educational technology, computer-aided design, software development, humanities computing, visualisation, games development and concurrent systems simulation. These are illustrated in the many models that can be found in the Empirical Modelling projects archive. They have also been the subject of graduate student research, as documented in the theses that can be accessed from the Publications link on the Empirical Modelling webpage.

There is scope to study the module in several ways, with many different kinds and levels of background experience. For the advanced student, there are good prospects for contributing to Empirical Modelling research through dissertation work. It is also possible to follow the module with little prior knowledge of computing. By way of context for the latter claim, consult the online *Sudoku Experience* workshops.

Page contact: Jackie Pinks Last revised: Tue 30 Nov 2010

UNIVERSITY OF WARWICK

<u>Proposal Form for New or Revised Modules (MA1- version 4)</u>

Approval information	
Approval Type	[✓ New module] ☐ Revised module ☐ Discontinue module
Date of Introduction/Change	01/10/2011
If new, does this module replace another? If so, enter module code and title:	NA
If revised/discontinued, please outline the rationale for the changes:	NA
Confirmation that affected departments have been consulted:	TBC

Module Summary	
1. Module Code (if known)	CS***
2. Module Title	Interdisciplinary Modelling through Constructivist Computing
3. Lead department:	Computer Science
4. Name of module leader	Meurig Beynon
5. Level	UG: ☐ Level 4 (Certificate) ☐ Level 5 (Intermediate) [✓ Level 6 (Honours)] PG: ☐ Level 7 (Masters) ☐ Level 8 (Doctoral) See Guidance Notes for relationship to years of study
6. Credit value(s) (CATS)	15
7. Principal Module Aims	This module will bring together students from different disciplines to address core themes through teamwork based on computer-based modelling. The principal

Module Summary	
	objective is to study ways in which computing technology can support such interdisciplinary modelling, and to highlight the merits of constructivist computing in this role. Constructivist computing is based on the principles, tools and potential applications of Empirical Modelling (EM), a new approach to modelling and computing that has been developed by staff and students here in Computer Science at Warwick over many years. Students will be instructed in fundamental EM concepts and techniques; be shown how EM is informed by perspectives relating to computer science, philosophy and constructivist thinking; be introduced to the fundamental notion of construal as it applies to key application areas; and participate in a collaborative modelling study directed at topics aligned to their personal interests and disciplinary specialisms.
8. Contact Hours (summary)	72
9. Assessment methods (summary)	Written report of Lab work - 25% Model analysis & documentation - 25% Group presentation - 25% Individual oral examination - 25%

Module Context

10. Please list all departments involved in the teaching of this module. If taught by more than one department, please indicate percentage split.

Computer Science

11. Availability of module

Degree Code	Title	Study Year	C/OC/ A/B/C	Credits
Example	Accounting and Finance International Summer School students	1 2 or 3	C	12 15

12. Minimum number of registered students required for module to run

8

13. Pre- and Post-Requisite Modules

NA

Module Content and Teaching

14. Teaching and Learning Activities

Lectures	10 (Week 1) + 10 (Week 2) + 2 external speakers
Seminars	5 seminars / demos on key models (weeks 2-3)
Tutorials	1 hour tutorial time for each student in total (20 minutes per week) with 3 students per tutorial session
Laboratory sessions	5 x 3 hours (Week 1) + 5 x 3 hours (Week 2) 3 x 3 hours (Week 3) These will comprise practical sessions involving construction and analysis together with group work (Week 3) Practical workshops relating to core themes (3 hrs) Group presentation sessions (2 hours)
Total contact hours	72
Module duration (weeks)	3
Other activity *(please describe): e.g. distance-learning, intensive weekend teaching etc.*	

15. Assessment Method (Standard)

Type of assessment	Length	% weighting

Module Content and Teaching		
Examinations	0 Hours	**0**
Assessed essays/coursework	Lab report / Model analysis + documentation - each equivalent to 2000 Words	**25% + 25%**
Other formal assessment	**Team presentation with individual contributions (2 hours)** **Individual oral exam (30 minutes)**	**25% + 25%**

16. Methods for providing feedback on assessment.

General feedback via module webpage/forum, individual feedback via tutorials. Detailed written feedback at the end of the module.

17. Outline Syllabus

The module will be delivered over three weeks, each week (comprising 5 days) representing a different phase of teaching and learning activity. In broad terms, Week 1 will motivate constructivist computing and introduce the basic principles, concepts and tools, Week 2 will be an in-depth study of papers and models relating to specific application areas for constructivist computing, and Week 3 will be devoted to group-based collaborative modelling focusing on one
of three core themes that will be advertised in advance of the module.

The portfolio of core themes for the module proposed for Summer 2012 is:

A. Modelling the musical and dramatic production and appreciation of Mozart's opera The Magic Flute. In this context, model-building could address the structural, thematic and harmonic aspects of the musical score, the characteristics of the cast (e.g. vocal range and physical characteristics) both as given and as ideally required, the layout of scenes and cast members involved in them, the design of costume and scenery, the movement of singers as planned by the artistic director, characteristics of specific performances such as relate to choice of tempi and potential cuts, the libretto and narrative, and the logistics (e.g.) of timetabling rehearsals and organising scene changes during a performance. Such a modelling exercise would draw on skills from computing, music, drama, business, design and education.

B. Making a model of a complex medical condition, such as HIV/AIDS, from multiple perspectives, in such a way as to be useful to trainee doctors, nursing staff with less specialist medical training, epidemiologists, patients, partners, and people without relevant medical knowledge. In this context, a model might simulate the case history of an AIDS patient according to current understanding of the pathophysiology of AIDS, taking account of the stages by which the HIV

Module Content and Teaching

infection impacts on a patient, adjusting their susceptibility to other infections, and assessing the risk of vertical transmission. A complementary analysis of the observables, dependencies and agencies at work would address factors that might influence the spread of the illness, such as the physical appearence of a patient and impact of cultural beliefs and customs. Relevant specialisms in this context include computing, medicine, education, statistics and social and anthropological studies.

C. Developing a suitable environment and associated suite of models to showcase the potential for using contructivist computing in teaching and learning mathematics and theoretical computer science. Different perspectives to be represented in this modelling might relate to the role being adopted by the model-builder (e.g. whether interaction is in the mode of developer, teacher or learner); what aspects of the curriculum are being targeted (e.g. basic mathematics for computer science, algorithmics, relational database theory, boolean circuit theory); the educational level (from elementary through intermediate to advanced level in school to undergraduate level in Computer Science and Mathematics); the focus (e.g. recreational mathematics, applied mathematics, mathematical research). Relevant areas of expertise in this context would include mathematics, computer science and education with particular emphasis on logical, computing, puzzle and problem-solving skills.

Each model-building study will be broad, have many aspects, and draw on several disciplines. In Week 3, each core theme will be addressed by a team of five students who will work alongside a Warwick Computer Science student with prior experience of EM in relevant areas of application. Though the quality of the models developed for specific aspects may not compete with what could be created using special-purpose commercial tools, the modelling has a quite distinctive characters and produces models unlike traditional computer models. Particularly significant is the flexibility, openness to reinterpretation and extension, and the way in which different aspects of the modelling are integrated and can be reconfigured.

Week 1 will introduce the notion of constructivist computing, as supported by Empirical Modelling (EM). The key notion of a construal will be introduced and discussed in relation to modelling and programming. The key concepts of observables, dependencies and agents will be explained and illustrated with standard examples. Practical model-building will be complemented by paper-based techniques for analysing complex systems based on classifying observables according to how they are perceived and manipulated by agents. The nature of applications characteristic of constructivist computing will be explored with reference to topics such as learning technologies, personalised applications, models for medical use, model-building to support experimental science or mathematical research, business modelling and humanities computing. An appropriate philosophical orientation for constructive computing will be briefly discussed with reference to critiques of the traditional "rationalistic" account of computing, Latour's proposals for rehabilitating the concept of construction in his paper "The Promises of Constructivism", and William James's radical empiricism.

In keeping with the spirit of constructivist computing, practical model building will be an essential strand of the module. The tools to be introduced will include the Cadence and EDEN interpreters and auxiliary environments such as the EM presentation environment (EMPE) and the dependency modelling tool (DMT).

Week 1 will involve ten lectures and ten laboratory sessions. Through personal consultations with students, we shall identify their particular strengths and interests and direct them towards different aspects of the core themes to be addressed in Week 3.

Week 2 will look in more detail at specific applications of EM relating to the chosen core themes, as represented by papers and model drawn from the EM archive. Each of the five days will be devoted to in-depth study of one or more existing models devoted to a specific theme (cf. the range of different kinds of observables, dependencies and agents represented in modelling strands associated with the core themes mentioned above, for which there are precedents in the EM archive in the form of construals of musical compositions, organs of the human body, timetabling models, mathematical algorithms and puzzles, visualisations and animations etc). Relevant key papers drawn from the EM archive will be studied in parallel. During this week, students will work on exercises in analysing and documenting models, typically making use of special-purpose tools such as the EMPE and the DMT.

In Week 2, there will be one or two introductory lectures and one or more sessions involving the analysis and demonstration of models relating to various aspects of the core themes. For the rest of the week, students will work in laboratory sessions alongside a small team of Warwick assistants (preferably undergraduates familiar with EM principles and tools who ideally would have themselves already worked on the modelling exercises and prototyped models related to the core themes).

Week 3 will be devoted to group work aimed at documenting and enhancing existing prototype models and conceiving and/or developing new models addressing the core themes. In the constructivist spirit, practical modelling activity will play a central role in this; even the conception of models will involve some experiments in model-building. An additional component of this activity that involves no conceptual shift within the EM framework might be enhancement of the modelling tools better to address the specific demands of a core theme.

The assessment for the module will be based on laboratory work in Week 1, the analysis and documentation exercise in Week 2, and the contribution to the group work in Week 3, as expressed through practical work and assessed through group presentations and individual oral examinations to be held at the end of Week 3.

Module Content and Teaching

To complement the above teaching programme, we shall organise one or two seminars from external speakers with specialist expertise relating to core themes, such as an expert in humanities computing or former EM doctoral students who have worked on (e.g.) collaborative group work using EM. We shall also engage other teaching assistants (e.g. from the Warwick Student Opera Group) to enact authentic activities relating to a core theme that help to illuminate the processes involved in the application (such as a "blocking rehearsal" for an ensemble from the Magic Flute, at which the precise movements and interactions of the singers are planned out in detail).

18. Illustrative Bibliography

Selected papers and models drawn from the EM publications and archived models at http://www.dcs.warwick.ac.uk/modelling
Fred Brooks, The Mythical Man Month Re-Visited, Addison-Wesley, 1995
Brian Cantwell-Smith, "The Foundations of Computing", In Scheutz, M.(ed) Computationalism: New Directions, MIT Press, p23.58, 2002
Chris Date and Hugh Darwen, Relational Database Writings (several books from 1985-1999), Addison-Wesley.
Paul Dourish, Where the Action is, MIT Press 2001
David Gooding, Experiment and the Making of Meaning: Human Agency in Scientific Observation and Experiment, Kluwer Academic, 1990
William James, Essays in Radical Empiricism, Bison Books 1996
Michael Jackson, "What can we expect of program verification?", IEEE Computer, 39(10):53{59, October 2006
David Harel and Rami Marelly, Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine, Springer-Verlag, 2003
William Kent, Data and Reality, 1st Books Library, 2000
Bruno Latour, "The Promises of Constructivism", In Ihde, D. (ed.) Chasing Technoscience: Matrix of Materiality, 2006
Willard McCarty, Humanities Computing, Palgrave-MacMillan, 2005
Drew McDermott, "A Critique of Pure Reason", Computer Intelligence 3 pp. 151-160 (1987) + subsequent responses in same journal.
Peter Naur, "Intuition in Software Development", TAPSOFT, Vol. 2, pages 60-79, 1985
Bonnie Nardi, A Small Matter of Programming, MIT Press, 1993.
Terry Winograd and Fernando Flores, Understanding Computers and Cognition: A New Foundation for Design, Addison-Wesley, 1986.

19. Learning outcomes

Successful completion of the module leads to the learning outcomes. The learning outcomes identify the knowledge, skills and attributes developed by the module.

Learning Outcomes should be presented in the format "By the end of the module students should be able to..." using the table at the end of the module approval form:

Resources

Resources
20. List any additional requirements and indicate the outcome of any discussions about these.

Approval	
21. Module leader's signature	
22. Date of approval	
23. Name of Approving Committee (include minute reference if applicable)	
24. Chair of Committee's signature	
25. Head of Department(s) Signature	

Examination Information		
A1. Name of examiner (if different from module leader)		
A2. Indicate all available methods of assessment in the table below		
% Examined	% Assessed by other methods	Length of examination paper
A3. Will this module be examined together with any other module (sectioned paper)? If so, please give details below.		
A4. How many papers will the module be examined by?	☐ 1 paper	☐ 2 papers
A5. When would you wish the exam take place (e.g. Jan, April, Summer)?		
A6. Is reading time required?	☐ Yes	☐ No
A7. Please specify any special exam timetable arrangements.		
A8. Stationery requirements		
No. of Answer books?		
Graph paper?		
Calculator?		
Any other special stationery requirements (e.g. Data books, tables etc)?		
A9. Type of examination paper		
Seen?	☐ Yes	☐ No
Open Book?	☐ Yes	☐ No
Restricted?	☐ Yes	☐ No

Examination Information	
If restricted, please provide a list of permitted texts:	

LEARNING OUTCOMES		
(By the end of the module the student should be able to....)	**Which teaching and learning methods enable students to achieve this learning outcome? (reference activities in section 15)**	**Which summative assessment method(s) will measure the achievement of this learning outcome? (reference activities in section 16)**
Appreciate the conceptual framework for interdisciplinary modelling that constructivist computing affords	Lectures and laboratory sessions Individual and group practical work.	All components of the assessment
Demonstrate familiarity with the basic principles of Empirical Modelling and the basic concepts of construals that embody patterns of observables, dependencies and agency	Lectures and laboratory sessions	Model analysis and documentation Lab report Individual oral examination
Understand, exercise and present standard existing EM construals	Lectures (especially those which incorporate model demonstrations) and laboratory sessions	Model analysis and documentation
Apply the principal tools and techniques deployed in constructivist computing	Laboratory sessions	Model analysis and documentation Lab report
Recognise the challenges presented by developing computer-based techniques and tools to support group work	Laboratory sessions devoted to group work	Group presentation Individual oral examination

Interdisciplinary Modelling through Constructivist Computing

A provisional budget for the module has been drawn up. Costs are accounted under three headings:

- Teaching costs (including preparation and assessment)

- Consumables

- Administrative / support infrastructure for the module

The costs are based on current university hourly rates for part-time teaching

The teaching costs total £6572.36 which can be broken down as follows:

Lectures and seminars	£1904.50
Student assistance	£3313.35
Staff lab demonstration	£ 469.17
Tutorials	£ 160.65
External contributors	£ 352.36
Additional activities	£ 132.33
Assessment	£ 240.00

	£6572.36

The consumables are costed at £269.50 and can be broken down as follows:

Photocopying materials for lectures	£ 100.00
Photocopying materials for labs	£ 72.00
Photocopying to support coursework	£ 22.50
Printing to support coursework	£ 75.00

	£ 269.50

The administrative and support costs can be broken down as follows:

Tool / model maintenance, webpage / forum	£ 198.50
Secretarial support	£ 225.00
Technical support	£ 300.00
Refreshments (Warwick Hospitality rates)	£ 780.00

	£1503.50

This makes an estimated overall cost of £8345.36.

(For more details of how these costs have been arrived at, see the appended annotated budget.)

```
## Lectures and seminars (20+5) in total 25 hours of specialist teaching

numlect = 25;
lectrate = 76.18;
lectures is numlect * lectrate;

## lectures = £1904.50

##############################

## student assistant support, assuming 3 undergraduate students
## distinguish assistance during the three weeks of the module (studM)
## from preparation activity to be conducted in summer of 2011 (studP)

studMrate = 11.31;
studM is studMrate * 15 * 3;
studPrate = 7.94;
studP is studPrate * 25 * 3;

student is studP + studM;
studN = 3;
students is student * studN;

## 3 students at £1104.45 = £3313.35

## students assistance on the module (studM) is 15 hours per week
## primarily looking after the daily 3 hours lab sessions
## there will also be a staff member supervising the lab session (demos)

demorate = 12.03;
numberlabs = 13;
hrsperlab = 3;
demos is demorate * numberlabs * hrsperlab;

## cost of staff member lab supervision is £469.17

################################

## module students will get in total 1 hour of tutorial time with lecturer
## tutorials will be in groups of three, probably cutting across team groupings
## 20 minutes per week with each of 5 tutorial groups, 5 hours tutorial in total

tuthrs = 5;
tutrate = 32.13;
tutorials = 5 * 32.13;

## cost of tutorials is £160.65

###################################

## External speakers giving guest seminar/lecture - travel+fee for 2 speakers

numexts = 2;
exttravel = 100;
extfee = 76.18;
extinput is numexts * ( exttravel + extfee);

## cost of external input is £352.36
## blocking rehearsal for the Magic Flute: 5 people for one hour at demo rate

addactMFhrs = 1;
addactMFstuds = 5;
addactMF is addactMFstuds * addactMFhrs * demorate;

## other activities: 2 hours involving (e.g.) 3 staff at demo rate

addactXhrs = 2;
addactXstaff = 3;
addactX is addactXhrs * addactXstaff * demorate;

addactivities is addactMF + addactX;

## total cost of additional activities is £132.33

###################################

## marking at 20 minutes per assessment item per student, charged at demo rate

markingRate is demorate;
totalmarkingtime = 1.33;
assessment is markingRate * numstuds * totalmarkingtime;

## assessment cost in total £240

###############################

## Total cost of the teaching components of the module:

teaching is students + demos + tutorials + lectures + extinput + addactivities +
assessment;

## Total teaching costs £6572.36

################################

## photocopying costs based on handouts of 4 sheets per lecture and lab session
## photocopying is charged at 5 pence per sheet

numstuds = 15;

lectnoteslen = 4;
lectnotescopies is numstuds + 5;
lectnotespersht = 5;
numlects = 25;
lectpccost is numlects * (lectnoteslen * lectnotescopies * lectnotespersht)/100.0;

## photocopies for lectures cost £100

labnoteslen = 4;
labnotescopies is numstuds + 5;
labnotespersht = 5;
numlabs = 18;
labpccost is numlabs * (labnoteslen * labnotescopies * labnotespersht)/100.0;

## photocopies for labs cost £72

photocopycost is lectpccost + labpccost;
```

```
## printing and copying costs for individual students
## at 5p per sheet, 100 printer credits, 30 photocopy credits
## takes account of resources generated in team work
## and (e.g.) papers cited for study in tutorials

printingcost is (numstuds * 5 * 100)/100.0;
indivphotocopycost is (numstuds * 5 * 30)/100.0;

## an additional £75 for printing and £22.50 for photocopying by individuals

consumables is photocopycost + printingcost + indivphotocopycost;

## consumables in total cost £269

################################

## assistance in tool/model maintenance, preparing & managing webpages/forum etc
## this could be partly during the module, partly in the week prior to the module
## equates to one 25 hours week for a student at lower rate

moduleadmin is studPrate * 25;

## tool/model maintenance, module webpage, forum etc administration costs £198.50

## 1 hour of secretarial assistance per day at £15 per hour- to include cost of
## (e.g.) memos and spreadsheets, photocopying, preparing & clearing refreshments

sechrsperday = 1;
secRate = 15;
secs is sechrsperday * secRate * 15;

## secretarial assistance costs £225

## 1 hour of technical per day
## lab technician / systems programmer at £20 per hour

techhrsperday = 1;
techRate = 20;
techs is techhrsperday * techRate * 15;

## technical assistance costs £300

## refreshments twice per day - tea and coffeee and cold drinks at £1.30 per person

tcwaterperhead = 1.30;
refreshments is tcwaterperhead * (numstuds + 5) * 15 * 2;

## refreshments cost £780 at Warwick Hospitality rates

envadmin is secs + techs + refreshments + moduleadmin;

## admin / systems cost of maintaining day to day environment is £1503.50

################################

totalcost is teaching + consumables + envadmin;

## overall cost £8345.36
```

29/2/2012

Preliminary enquiry from Meurig Beynon regarding proposed ISS proposal

Background

The university established its International Summer School (ISS) for undergraduate students last year. It is a part of the university strategy. With reference to the university's documentation: "The aim of the Summer School was to showcase the best of Warwick to non-traditional Warwick students who were part-way through their degree course, but had not yet graduated". It would be beneficial to the department to be involved in this initiative both from an internal and an external perspective - it might be a good way in which to raise the departmental profile internationally both at undergraduate and postgraduate levels. My purpose here is to seek provisional departmental support for an ISS proposal.

The first modules on the ISS programme will be offered in July 2012. Together with Steve Russ, I proposed a module last year. This would have been scheduled to run in 2012 but for my failure to gain the support of the Computer Science department when the proposal was presented for approval at the Science Faculty Board. After reflecting on last year's experience, I've been considering preparing another - hopefully more appropriately focused - proposal for an ISS module to be launched in the summer of 2013. The new proposal is on a theme much more closely aligned to mainstream computer science: that of "understanding computer programming" (provisional draft of content attached). From what I know of ISS modules that have been proposed by other departments, it is common for them to be led by an Emeritus member of staff, and - if it is possible to get the requisite moral and practical support from departmental colleagues and research students - I would be more than happy to take on the role of module leader and principal contributor.

An outline proposal

An ISS module runs for 3 weeks and with 15 or so international undergraduate students with 2-3 years of experience of university study. The students earn credit from the module that they can take back to their home institution - the ISS modules already approved will count for 15 CATS.

As a rough indication of what I have in mind for the module, about half of the total time would be devoted to workshop-style sessions on mainstream topics such as Turing machines, object-oriented programming (as in an introductory Java or Greenfoot), functional programming, software specification, databases, and software for educational use (e.g. NetLogo, Scratch). This material would be complemented by similar sessions in which these themes were considered from an EM perspective. Other programming-related topics (such as high-performance computing, dependable systems, agent-oriented modelling) might also feature in guest lectures, as would contributions from former EM research students with industrial experience of software development. Credit would be gained primarily through the practical programming-related exercises associated with the workshops.

Resources

The mainstream computer science material could be drawn from existing computer science modules. The aim in each area would be to focus on ideas and practical exercises that are conceptually interesting where 'understanding computer programming' is concerned. The EM material already exists for the most part in lectures and illustrative models presented in the CS405 module. The bibliography for the module would comprise 10-15 key papers (e.g. Turing, Fred Brooks, David Harel, Bonnie Nardi, Peter Naur, Michael Jackson etc) and a similar number of EM publications.

For 2012, the University was prepared to fund each module up to 8K pounds to cover preparation and delivery costs. I would not myself need to be paid for my part in teaching on the module and envisage using such a budget entirely to support other preparation, teaching, technical and administrative demands. Subject to consultation with academics, RAs could be used in this role.

Understanding computer programming (working title)

Computer programming is at the heart of computer science. Warwick Computer Science has a strong and well-established tradition of research in this area. It has been a leading international contributor to building on the algorithmic and logical foundations for computer science laid in the work of Turing. Over the years, Warwick CS has been at the forefront of research in matters associated with paradigms and principles of programming, programming language design and semantics, and specialised forms of programming for practical applications.

Perhaps surprisingly, in view of the vast investment of intellectual effort in programming over the last 50 years, many aspects of programming remain conceptually challenging and problematic in practice. There is evidence for this in 'the software crisis' - as manifest in the well-publicised failures of high-profile IT projects. Though diverse programming paradigms and modes of programming specification have been developed, it is not straightforward to make these coherent or integrate them effectively in practice. Visions for human-machine symbiosis, end-user programming, automated modes of programming based on learning and evolutionary models have been around for decades, but have yet to be realised. Technological developments, such as multi-processors, multi-media and web architectures, coupled with modes of use that feature embedding and embodiment, continue to extend the boundaries of 'programming' in the wild. In computer science education, despite the invention of many hundred varieties of Logo for instance, the question of what programming languages are suitable for introducing computing at school remains unresolved. The relationship between computer science and 'programming' as it is represented in Information and Communication Technologies (ICT) (as in the use of spreadsheets, databases and special-purpose applications e.g. for CAD, animation and music synthesis) is controversial and unclear.

Empirical Modelling (EM), as developed in Computer Science at Warwick over the last 25 years, provides a broader conceptual framework within which to study computer programming, together with principles and tools that make it possible to approach programming from a fresh and illuminating perspective. Whereas classical programming (as represented in algorithmics and PL design and semantics) focuses on the relationship between a program and the behaviour it induces in the machine, the primary focus in EM is on understanding how this machine behaviour relates to the real-world context in which it is embedded. Such a shift in focus necessarily draws attention to the relationship between formal and intuitive aspects of mathematics: a theme highlighted in the work of the mathematician Emil Post (of the Post Correspondence Problem), and further elaborated in relation to programming practice from many different perspectives by (e.g.) Brian Cantwell-Smith, Peter Naur, Winograd and Flores, Bonnie Nardi, Michael Jackson and Willard McCarty.

A key element in rethinking programming within EM is putting the primary emphasis on the immediate experience of the programmer. Rather than specifying behaviours in a tightly constrained context, the programmer's attention is directed at developing a flexible interactive context in which rich kinds of agency can be enacted. This mode of development is in line with the use of spreadsheets, web development, and particular varieties of programming practice where visualisation and/or the user experience has a critical role, as in computer games. A distinctive feature of EM that has been prominent from the beginning is the use of dependency maintenance such as is now becoming a feature of commercial PLs (Flex, Microsoft WPF, JavaScript). The fact that introducing dependency to traditional programming environments exacerbates the problems of incoherence is itself a strong motivation for the radical reconceptualisation that underpins EM.

The broad aim of this module would be to present a perspective on computer programming that draws on the expertise of Warwick Computer Science in key mainstream areas and the complementary commentary, critique and reconstruction that EM offers.

Understanding computer programming (working title)

Computer programming is at the heart of computer science. Warwick Computer Science has a strong and well-established tradition of research in this area. It has been a leading international contributor to building on the algorithmic and logical foundations for computer science laid in the work of Turing. Over the years, Warwick CS has been at the forefront of research in matters associated with paradigms and principles of programming, programming language design and semantics, and specialised forms of programming for practical applications.

Perhaps surprisingly, in view of the vast investment of intellectual effort in programming over the last 50 years, many aspects of programming remain conceptually challenging and problematic in practice. There is evidence for this in 'the software crisis' - as manifest in the well-publicised failures of high-profile IT projects. Though diverse programming paradigms and modes of programming specification have been developed, it is not straightforward to make these coherent or integrate them effectively in practice. Visions for human-machine symbiosis, end-user programming, automated modes of programming based on learning and evolutionary models have been around for decades, but have yet to be realised. Technological developments, such as multi-processors, multi-media and web architectures, coupled with modes of use that feature embedding and embodiment, continue to extend the boundaries of 'programming' in the wild. In computer science education, despite the invention of many hundred varieties of Logo for instance, the question of what programming languages are suitable for introducing computing at school remains unresolved. The relationship between computer science and 'programming' as it is represented in Information and Communication Technologies (ICT) (as in the use of spreadsheets, databases and special-purpose applications e.g. for CAD, animation and music synthesis) is controversial and unclear.

Empirical Modelling (EM), as developed in Computer Science at Warwick over the last 25 years, provides a broader conceptual framework within which to study computer programming, together with principles and tools that make it possible to approach programming from a fresh and illuminating perspective. Whereas classical programming (as represented in algorithmics and PL design and semantics) focuses on the relationship between a program and the behaviour it induces in the machine, the primary focus in EM is on understanding how this machine behaviour relates to the real-world context in which it is embedded. Such a shift in focus necessarily draws attention to the relationship between formal and intuitive aspects of mathematics: a theme highlighted in the work of the mathematician Emil Post (of the Post Correspondence Problem), and further elaborated in relation to programming practice from many different perspectives by (e.g.) Brian Cantwell-Smith, Peter Naur, Winograd and Flores, Bonnie Nardi, Michael Jackson and Willard McCarty.

A key element in rethinking programming within EM is putting the primary emphasis on the immediate experience of the programmer. Rather than specifying behaviours in a tightly constrained context, the programmer's attention is directed at developing a flexible interactive context in which rich kinds of agency can be enacted. This mode of development is in line with the use of spreadsheets, web development, and particular varieties of programming practice where visualisation and/or the user experience has a critical role, as in computer games. A distinctive feature of EM that has been prominent from the beginning is the use of dependency maintenance such as is now becoming a feature of commercial PLs (Flex, Microsoft WPF, JavaScript). The fact that introducing dependency to traditional programming environments exacerbates the problems of incoherence is itself a strong motivation for the radical reconceptualisation that underpins EM.

The broad aim of this module would be to present a perspective on computer programming that draws on the expertise of Warwick Computer Science in key mainstream areas and the complementary commentary, critique and reconstruction that EM offers.